Comprehensive Analysis of NBA Draft Trends Evaluating Career Longevity, Performance Metrics, and Success of Top 3 Picks

By Shaun McKellar Jr

Import Packages and Load Dataset

In [1]:
```
import nbformat
from nbconvert import PDFExporter
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm

# Load the dataset
file_path = '/Users/shaunmckellarjr/Desktop/Shaun_Projects/NBA Draft Data/nb
nba_draft_data = pd.read_csv(file_path)
```

Inspect Data

In [2]:
```
print(nba_draft_data.head())
```

```
   id  year  rank  overall_pick team          player     college  \
0   1  1989     1             1  SAC  Pervis Ellison  Louisville
1   2  1989     2             2  LAC     Danny Ferry        Duke
2   3  1989     3             3  SAS    Sean Elliott     Arizona
3   4  1989     4             4  MIA       Glen Rice    Michigan
4   5  1989     5             5  CHH       J.R. Reid         UNC

   years_active   games  minutes_played  ...  3_point_percentage  \
0          11.0   474.0         11593.0  ...               0.050
1          13.0   917.0         18133.0  ...               0.393
2          12.0   742.0         24502.0  ...               0.375
3          15.0  1000.0         34985.0  ...               0.400
4          11.0   672.0         15370.0  ...               0.135

   free_throw_percentage  average_minutes_played  points_per_game  \
0                  0.689                    24.5              9.5
1                  0.840                    19.8              7.0
2                  0.799                    33.0             14.2
3                  0.846                    35.0             18.3
4                  0.716                    22.9              8.5

   average_total_rebounds  average_assists  win_shares  \
0                     6.7              1.5        21.8
1                     2.8              1.3        34.8
2                     4.3              2.6        55.7
3                     4.4              2.1        88.7
4                     5.0              1.0        22.5

   win_shares_per_48_minutes  box_plus_minus  value_over_replacement
0                      0.090            -0.5                     4.4
1                      0.092            -0.9                     4.9
2                      0.109             0.2                    13.5
3                      0.122             0.8                    24.9
4                      0.070            -2.9                    -3.7

[5 rows x 24 columns]
```

Summary of Data

In [3]:
```
print(nba_draft_data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1922 entries, 0 to 1921
Data columns (total 24 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   id                        1922 non-null   int64  
 1   year                      1922 non-null   int64  
 2   rank                      1922 non-null   int64  
 3   overall_pick              1922 non-null   int64  
 4   team                      1922 non-null   object 
 5   player                    1922 non-null   object 
 6   college                   1585 non-null   object 
 7   years_active              1669 non-null   float64
 8   games                     1669 non-null   float64
 9   minutes_played            1669 non-null   float64
 10  points                    1669 non-null   float64
 11  total_rebounds            1669 non-null   float64
 12  assists                   1669 non-null   float64
 13  field_goal_percentage     1665 non-null   float64
 14  3_point_percentage        1545 non-null   float64
 15  free_throw_percentage     1633 non-null   float64
 16  average_minutes_played    1669 non-null   float64
 17  points_per_game           1669 non-null   float64
 18  average_total_rebounds    1669 non-null   float64
 19  average_assists           1669 non-null   float64
 20  win_shares                1669 non-null   float64
 21  win_shares_per_48_minutes 1668 non-null   float64
 22  box_plus_minus            1668 non-null   float64
 23  value_over_replacement    1669 non-null   float64
dtypes: float64(17), int64(4), object(3)
memory usage: 360.5+ KB
None
```

Check for missing values

In [4]:
```
print(nba_draft_data.isnull().sum())
```

```
id                          0
year                        0
rank                        0
overall_pick                0
team                        0
player                      0
college                   337
years_active              253
games                     253
minutes_played            253
points                    253
total_rebounds            253
assists                   253
field_goal_percentage     257
3_point_percentage        377
free_throw_percentage     289
average_minutes_played    253
points_per_game           253
average_total_rebounds    253
average_assists           253
win_shares                253
win_shares_per_48_minutes 254
box_plus_minus            254
value_over_replacement    253
dtype: int64
```

Descriptive Statistics

In [5]:
```
# Get basic descriptive statistics
print(nba_draft_data.describe())
```

	id	year	rank	overall_pick	years_active
count	1922.000000	1922.000000	1922.000000	1922.000000	1669.000000
mean	961.500000	2005.317378	29.694589	29.694589	6.332534
std	554.977927	9.456946	16.912454	16.912454	4.656321
min	1.000000	1989.000000	1.000000	1.000000	1.000000
25%	481.250000	1997.000000	15.000000	15.000000	2.000000
50%	961.500000	2005.000000	30.000000	30.000000	5.000000
75%	1441.750000	2013.000000	44.000000	44.000000	10.000000
max	1922.000000	2021.000000	60.000000	60.000000	22.000000

	games	minutes_played	points	total_rebounds
count	1669.000000	1669.000000	1669.000000	1669.000000
mean	348.042540	8399.055722	3580.413421	1497.009587
std	324.897567	9845.871529	4826.142847	2003.686388
min	1.000000	0.000000	0.000000	0.000000
25%	72.000000	838.000000	265.000000	128.000000
50%	235.000000	4204.000000	1552.000000	656.000000
75%	584.000000	13246.000000	5150.000000	2139.000000
max	1541.000000	52139.000000	37062.000000	15091.000000

	assists	...	3_point_percentage	free_throw_percentage
count	1669.000000	...	1545.000000	1633.000000
mean	774.300779	...	0.272405	0.716825
std	1284.602969	...	0.128339	0.118702
min	0.000000	...	0.000000	0.000000
25%	46.000000	...	0.222000	0.659000
50%	257.000000	...	0.317000	0.736000
75%	910.000000	...	0.356000	0.797000
max	12091.000000	...	1.000000	1.000000

	average_minutes_played	points_per_game	average_total_rebounds
count	1669.000000	1669.000000	1669.000000
mean	18.134032	7.275734	3.194368
std	8.707656	4.969343	2.083895
min	0.000000	0.000000	0.000000
25%	11.000000	3.400000	1.700000
50%	17.700000	6.200000	2.800000
75%	24.800000	10.000000	4.200000
max	41.100000	27.200000	13.300000

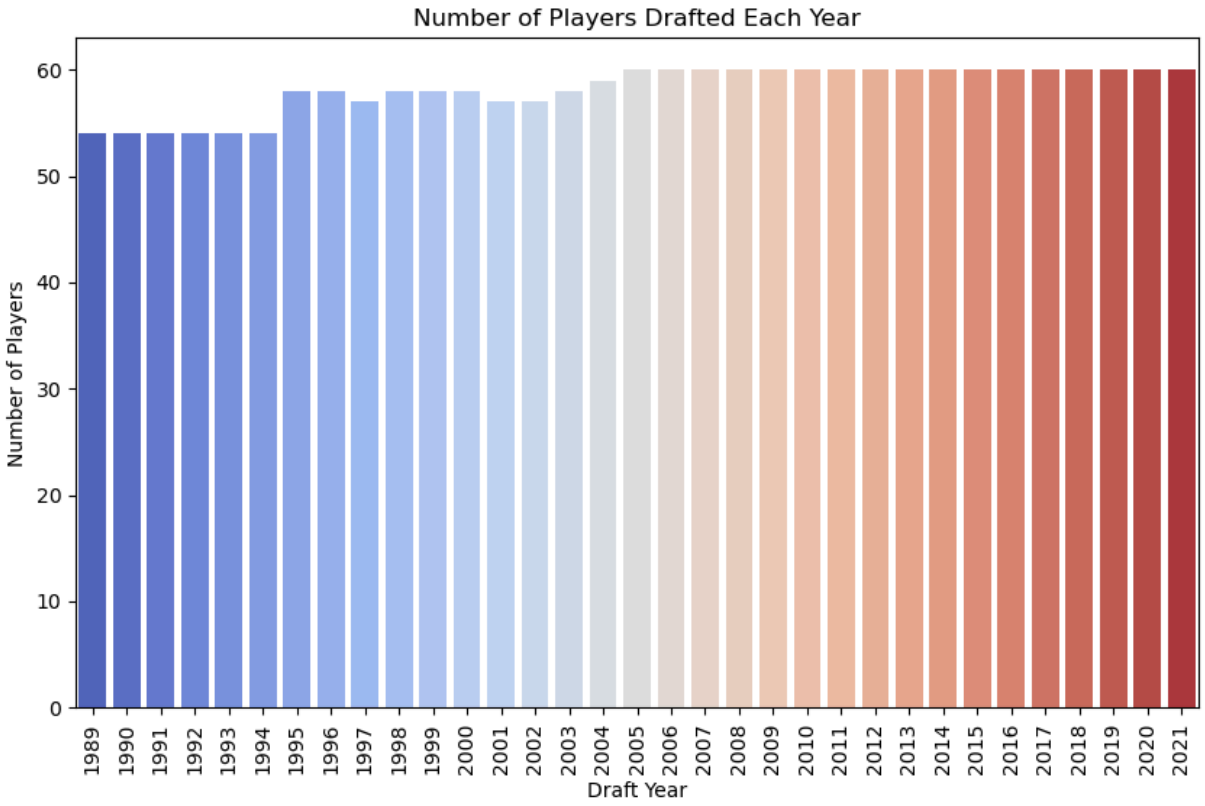
	average_assists	win_shares	win_shares_per_48_minutes
count	1669.000000	1669.000000	1668.000000
mean	1.550749	17.873697	0.061691
std	1.488536	27.989805	0.094467
min	0.000000	-1.700000	-1.264000
25%	0.500000	0.400000	0.030000
50%	1.100000	5.300000	0.069000
75%	2.100000	24.500000	0.104000
max	9.500000	249.500000	1.442000

	box_plus_minus	value_over_replacement
count	1668.000000	1669.000000
mean	-2.311271	4.403176
std	4.143403	11.461729
min	-52.000000	-8.500000
25%	-3.900000	-0.400000

```
50%          -2.000000                0.000000
75%          -0.300000                4.500000
max          51.100000              142.600000

[8 rows x 21 columns]
```

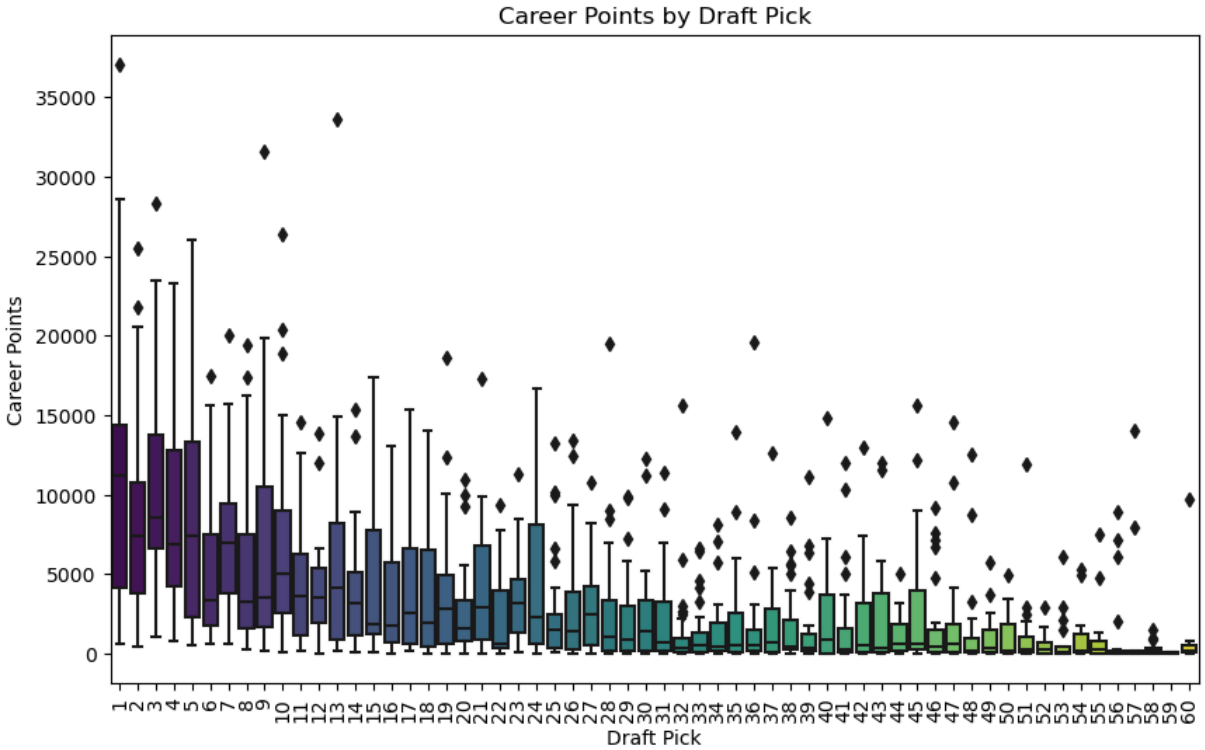
Number of Players Drafted Per Year

In [6]:
```
plt.figure(figsize=(10, 6))
ax = sns.countplot(data=nba_draft_data, x='year', palette='coolwarm')
plt.title('Number of Players Drafted Each Year')
plt.xlabel('Draft Year')
plt.ylabel('Number of Players')
plt.xticks(rotation=90)

plt.show()
```



Career Points by Draft Pick

In [7]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(data=nba_draft_data, x='overall_pick', y='points', palette='viri
plt.title('Career Points by Draft Pick')
plt.xlabel('Draft Pick')
plt.ylabel('Career Points')
plt.xticks(rotation=90)
plt.show()
```



In [8]:
```
# Ensure relevant columns are in the correct data type
nba_draft_data['overall_pick'] = nba_draft_data['overall_pick'].astype(int)
nba_draft_data['years_active'] = nba_draft_data['years_active'].astype(float
```

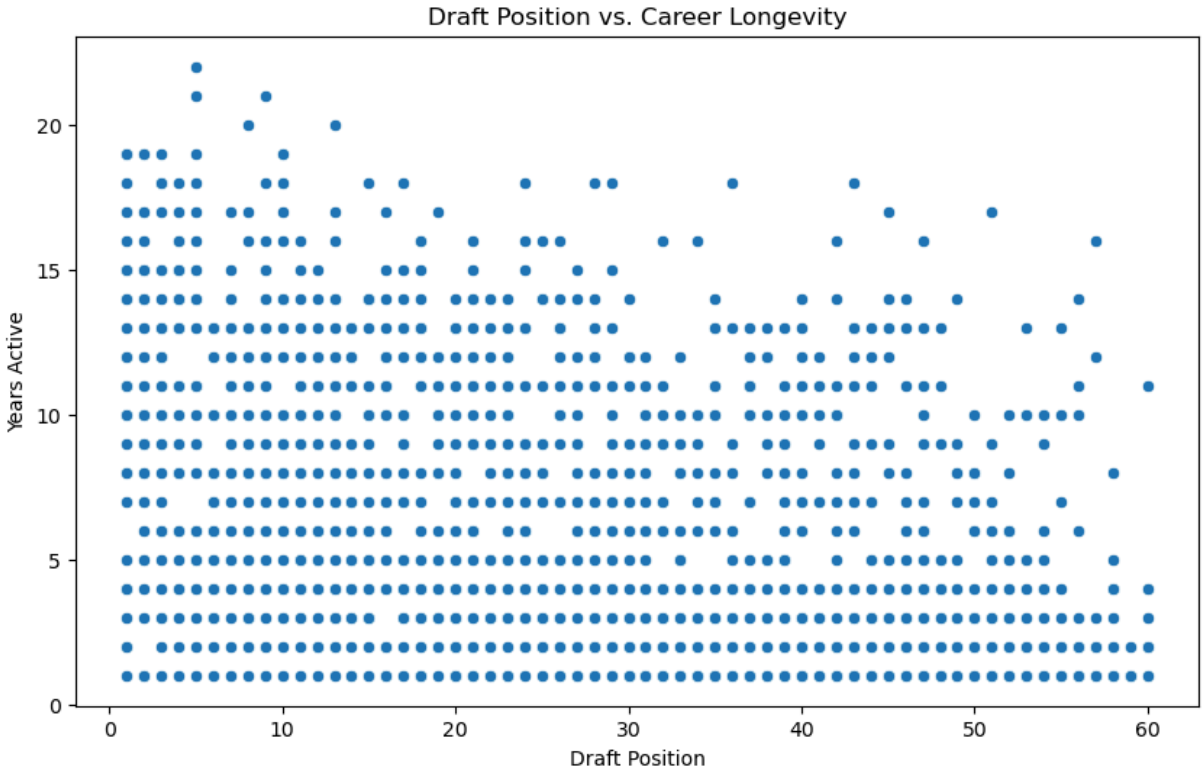
In [9]:
```
# Drop rows with missing values in the relevant columns
nba_draft_data = nba_draft_data.dropna(subset=['overall_pick', 'years_active
```

In [10]:
```
# Compute the correlation
correlation = nba_draft_data[['overall_pick', 'years_active']].corr()
print(correlation)
```

```
              overall_pick  years_active
overall_pick       1.00000      -0.42805
years_active      -0.42805       1.00000
```

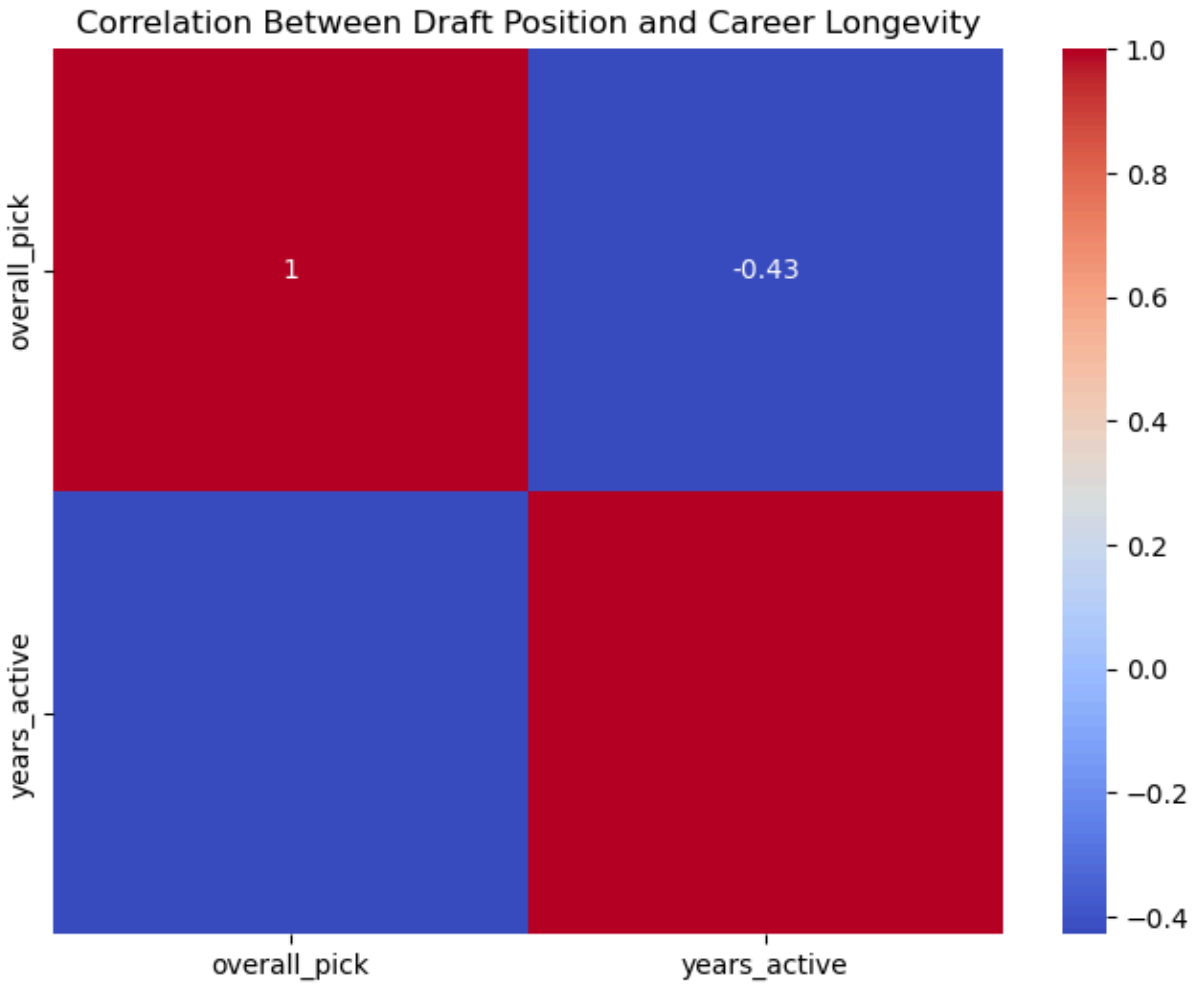
Scatter plot to visualize Draft Position vs Career Longevity

In [11]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=nba_draft_data, x='overall_pick', y='years_active')
plt.title('Draft Position vs. Career Longevity')
plt.xlabel('Draft Position')
plt.ylabel('Years Active')
plt.show()
```



Correlation heatmap that showcases Between Draft Position and Career Longevity

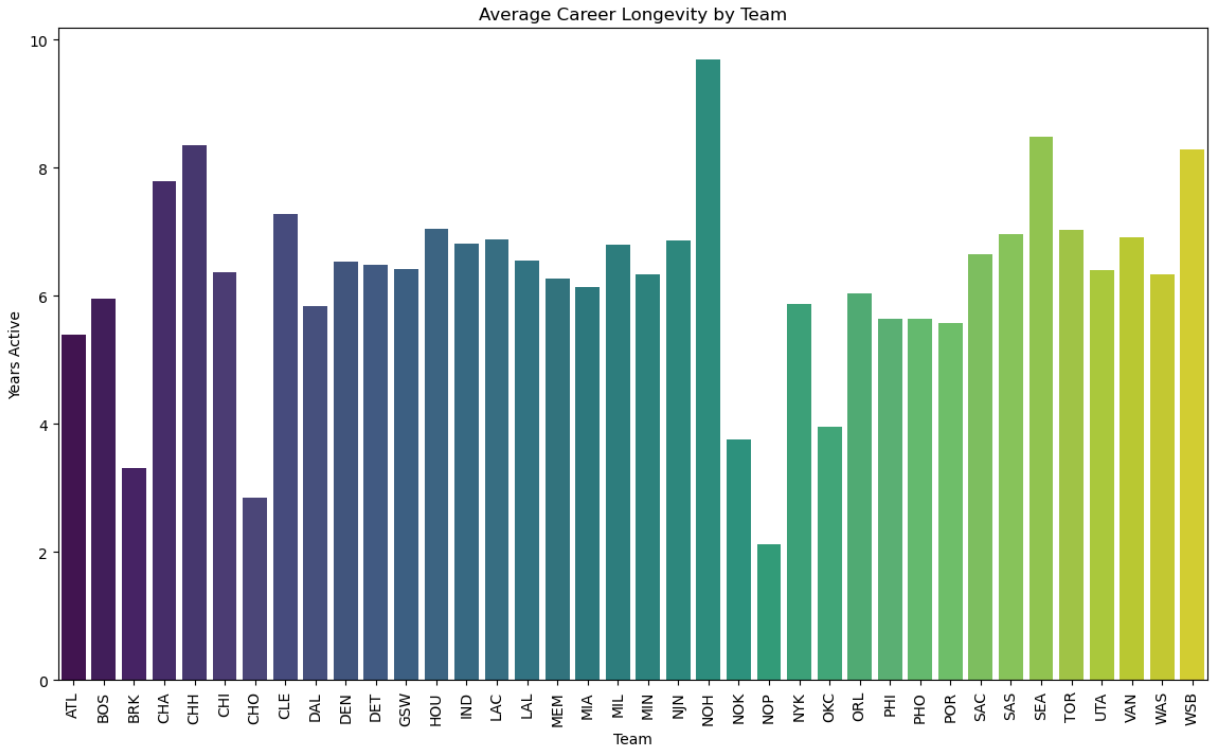
In [12]:
```
# Correlation heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation, annot=True, cmap='coolwarm')
plt.title('Correlation Between Draft Position and Career Longevity')
plt.show()
```

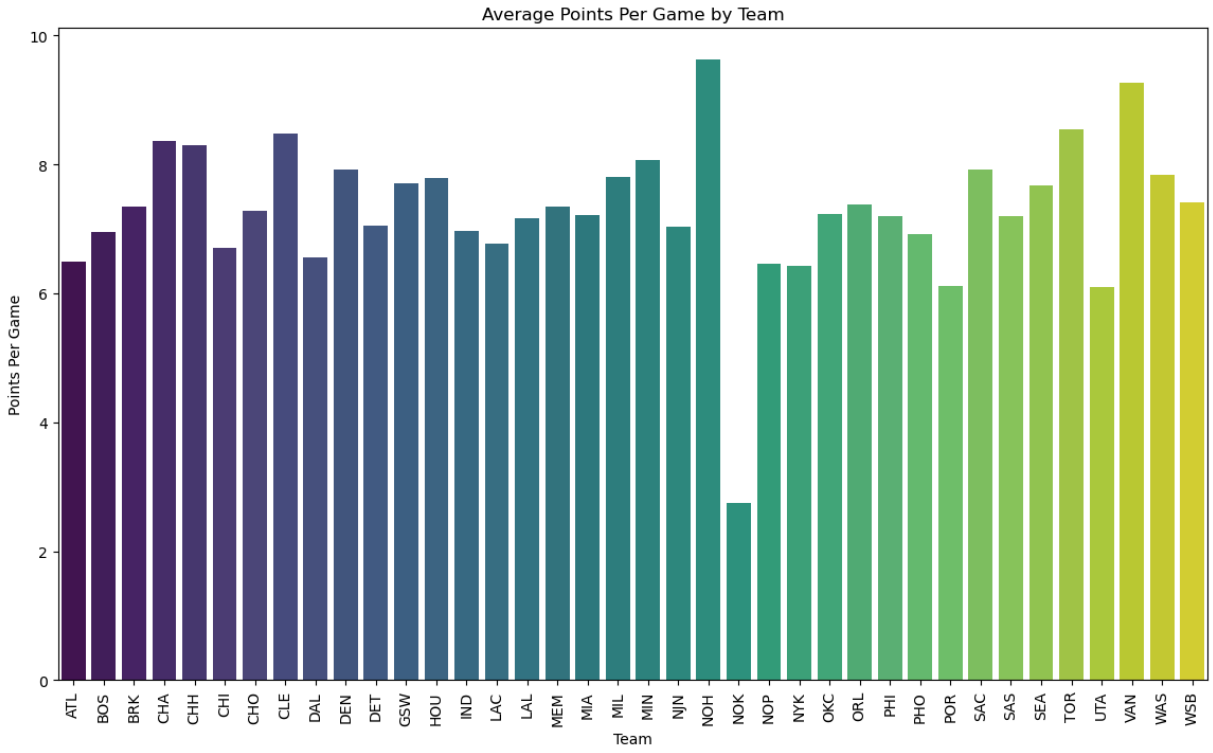


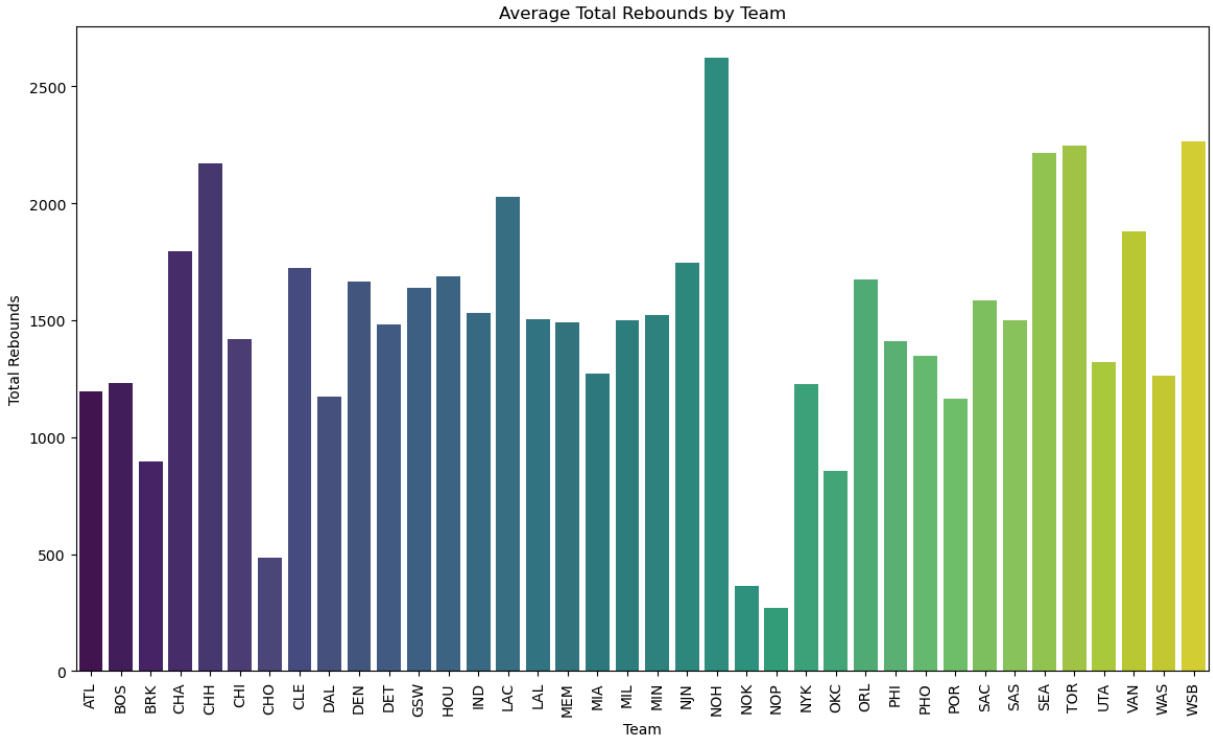
Avg performance metrics for numeric columns only

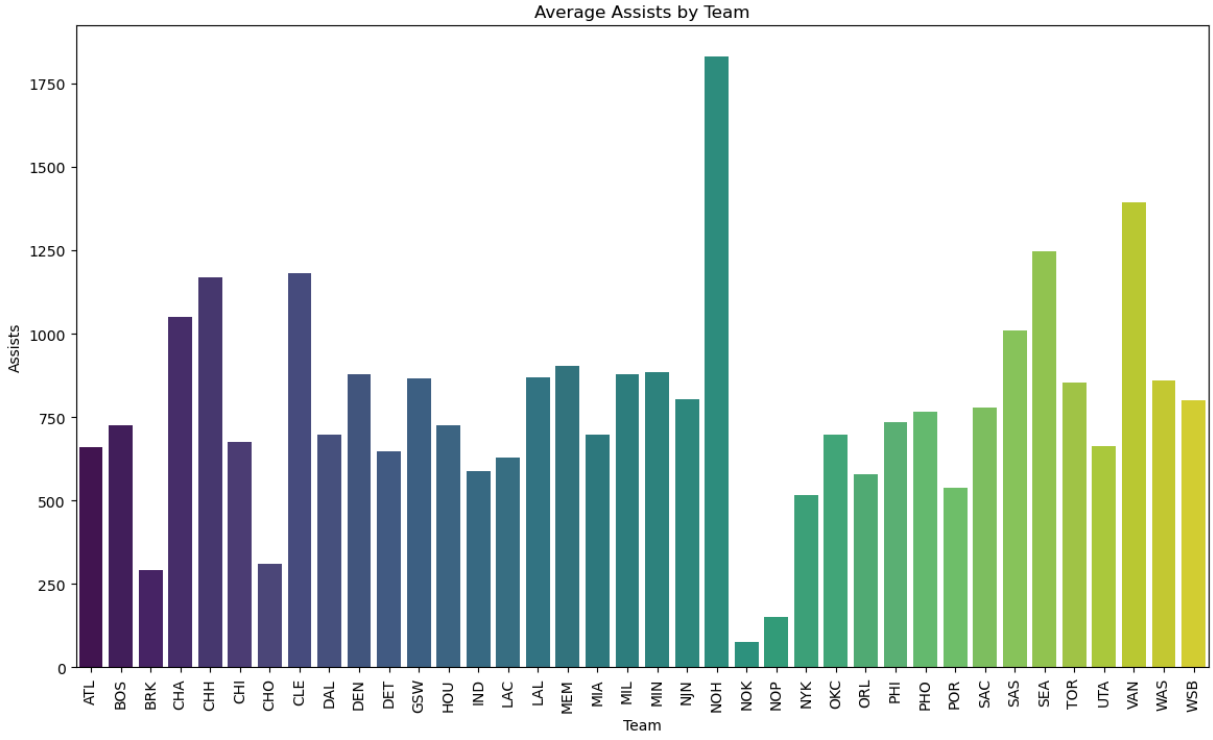
In [13]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Select relevant features for the analysis
features = ['team', 'player', 'years_active', 'points_per_game', 'total_rebo

# Filter the data to include only the relevant features
nba_draft_data_filtered = nba_draft_data[features]

# Group by team and calculate mean performance metrics for numeric columns o
numeric_columns = ['years_active', 'points_per_game', 'total_rebounds', 'ass
team_summary = nba_draft_data_filtered.groupby('team')[numeric_columns].mear

# Plot career longevity by team
plt.figure(figsize=(14, 8))
sns.barplot(data=team_summary, x='team', y='years_active', palette='viridis'
plt.title('Average Career Longevity by Team')
plt.xlabel('Team')
plt.ylabel('Years Active')
```

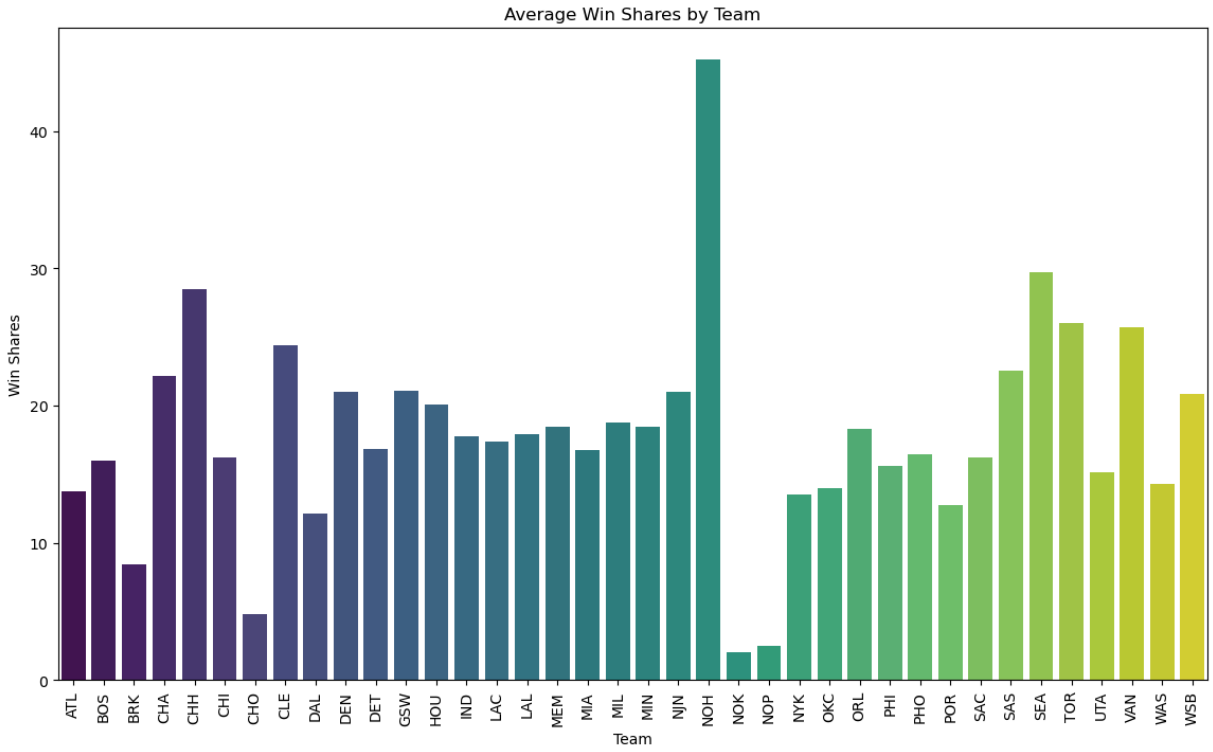
```
plt.xticks(rotation=90)
plt.show()

# Plot points per game by team
plt.figure(figsize=(14, 8))
sns.barplot(data=team_summary, x='team', y='points_per_game', palette='viric
plt.title('Average Points Per Game by Team')
plt.xlabel('Team')
plt.ylabel('Points Per Game')
plt.xticks(rotation=90)
plt.show()

# Plot total rebounds by team
plt.figure(figsize=(14, 8))
sns.barplot(data=team_summary, x='team', y='total_rebounds', palette='viridi
plt.title('Average Total Rebounds by Team')
plt.xlabel('Team')
plt.ylabel('Total Rebounds')
plt.xticks(rotation=90)
plt.show()

# Plot assists by team
plt.figure(figsize=(14, 8))
sns.barplot(data=team_summary, x='team', y='assists', palette='viridis')
plt.title('Average Assists by Team')
plt.xlabel('Team')
plt.ylabel('Assists')
plt.xticks(rotation=90)
plt.show()

# Plot win shares by team
plt.figure(figsize=(14, 8))
sns.barplot(data=team_summary, x='team', y='win_shares', palette='viridis')
plt.title('Average Win Shares by Team')
plt.xlabel('Team')
plt.ylabel('Win Shares')
plt.xticks(rotation=90)
plt.show()
```


Average Career Longevity by Team
Years Active
0
2
4
6
8
10
ATL BOS BRK CHA CHH CHI CHO CLE DAL DEN DET GSW HOU IND LAC LAL MEM MIA MIL MIN NJN NOH NOK NOP NYK OKC ORL PHI PHO POR SAC SAS SEA TOR UTA VAN WAS WSB
Team


Average Points Per Game by Team
Points Per Game
0
2
4
6
8
10
ATL BOS BRK CHA CHH CHI CHO CLE DAL DEN DET GSW HOU IND LAC LAL MEM MIA MIL MIN NJN NOH NOK NOP NYK OKC ORL PHI PHO POR SAC SAS SEA TOR UTA VAN WAS WSB
Team


Average Total Rebounds by Team
Total Rebounds
0
500
1000
1500
2000
2500
ATL BOS BRK CHA CHH CHI CHO CLE DAL DEN DET GSW HOU IND LAC LAL MEM MIA MIL MIN NJN NOH NOK NOP NYK OKC ORL PHI PHO POR SAC SAS SEA TOR UTA VAN WAS WSB
Team


Average Assists by Team
Assists
0
250
500
750
1000
1250
1500
1750
ATL BOS BRK CHA CHH CHI CHO CLE DAL DEN DET GSW HOU IND LAC LAL MEM MIA MIL MIN NJN NOH NOK NOP NYK OKC ORL PHI PHO POR SAC SAS SEA TOR UTA VAN WAS WSB
Team



Late picks with long careers

In [14]:
```
# Check outliers: Late picks with long careers
late_picks_long_careers = nba_draft_data[(nba_draft_data['overall_pick'] > 3
print(late_picks_long_careers)
```

```
        id  year  rank  overall_pick team            player        college
\
35      36  1989    36            36  POR  Clifford Robinson          UConn
37      38  1989    38            38  MIN          Doug West      Villanova
42      43  1989    43            43  CLE       Chucky Brown       NC State
88      89  1990    35            35  WSB        Greg Foster  Texas-El Paso
91      92  1990    38            38  SEA       Jud Buechler        Arizona
...    ...   ...   ...           ...  ...                ...            ...
1118  1119  2008    37            37  MIL   Luc Mbah a Moute          UCLA
1126  1127  2008    45            45  SAS       Goran Dragić           NaN
1187  1188  2009    46            46  CLE        Danny Green           UNC
1196  1197  2009    55            55  POR        Patty Mills   Saint Mary's
1321  1322  2011    60            60  SAC      Isaiah Thomas     Washington

      years_active   games  minutes_played  ...  3_point_percentage  \
35            18.0  1380.0         42561.0  ...               0.356
37            12.0   676.0         16649.0  ...               0.191
42            13.0   694.0         11962.0  ...               0.227
88            13.0   656.0          7974.0  ...               0.225
91            12.0   720.0          8444.0  ...               0.366
...            ...     ...             ...  ...                 ...
1118          12.0   689.0         16038.0  ...               0.334
1126          14.0   888.0         24734.0  ...               0.362
1187          13.0   819.0         20743.0  ...               0.399
1196          13.0   820.0         17013.0  ...               0.389
1321          11.0   550.0         15563.0  ...               0.362

      free_throw_percentage  average_minutes_played  points_per_game  \
35                    0.689                    30.8             14.2
37                    0.801                    24.6              9.6
42                    0.699                    17.2              5.9
88                    0.748                    12.2              3.9
91                    0.633                    11.7              3.3
...                     ...                     ...              ...
1118                  0.659                    23.3              6.4
1126                  0.767                    27.9             13.7
1187                  0.804                    25.3              8.7
1196                  0.855                    20.7              9.2
1321                  0.872                    28.3             17.7

      average_total_rebounds  average_assists  win_shares  \
35                       4.6              2.2        89.7
37                       2.5              1.9        17.7
42                       3.1              0.8        19.0
88                       2.6              0.5         5.6
91                       1.8              0.8        16.5
...                      ...              ...         ...
1118                     4.1              0.9        26.0
1126                     3.1              4.8        56.3
1187                     3.4              1.6        49.1
1196                     1.7              2.3        35.3
1321                     2.4              4.8        45.3

      win_shares_per_48_minutes  box_plus_minus  value_over_replacement
35                        0.101             0.5                    26.6
37                        0.051            -3.0                    -4.2
```

```
42                      0.076          -2.7                -2.0
88                      0.033          -5.5                -7.0
91                      0.094          -0.1                 4.0
...                       ...           ...                 ...
1118                    0.078          -1.5                 2.0
1126                    0.109           0.6                16.3
1187                    0.114           1.9                20.2
1196                    0.100           0.1                 9.0
1321                    0.140           2.1                16.1

[65 rows x 24 columns]
```

In [15]:
```
# Define the dependent and independent variables
X = nba_draft_data[['overall_pick']]
y = nba_draft_data['years_active']

# Add a constant to the model (intercept)
X = sm.add_constant(X)

# Fit the model
model = sm.OLS(y, X).fit()

# Print the summary
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
==
Dep. Variable:           years_active   R-squared:                       0.1
83
Model:                            OLS   Adj. R-squared:                  0.1
83
Method:                 Least Squares   F-statistic:                     37
4.0
Date:                Thu, 30 May 2024   Prob (F-statistic):           2.48e-
75
Time:                        12:57:39   Log-Likelihood:                -476
6.1
No. Observations:                1669   AIC:                             953
6.
Df Residuals:                    1667   BIC:                             954
7.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
====
                 coef    std err          t      P>|t|      [0.025      0.
975]
------------------------------------------------------------------------------
----
const          9.6668      0.201     48.126      0.000       9.273      1
0.061
overall_pick    -0.1240      0.006    -19.338      0.000      -0.137      -
0.111
==============================================================================
==
Omnibus:                      105.910   Durbin-Watson:                   1.5
00
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              124.6
07
Skew:                           0.662   Prob(JB):                     8.75e-
28
Kurtosis:                       2.809   Cond. No.                         6
1.1
==============================================================================
==

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
```

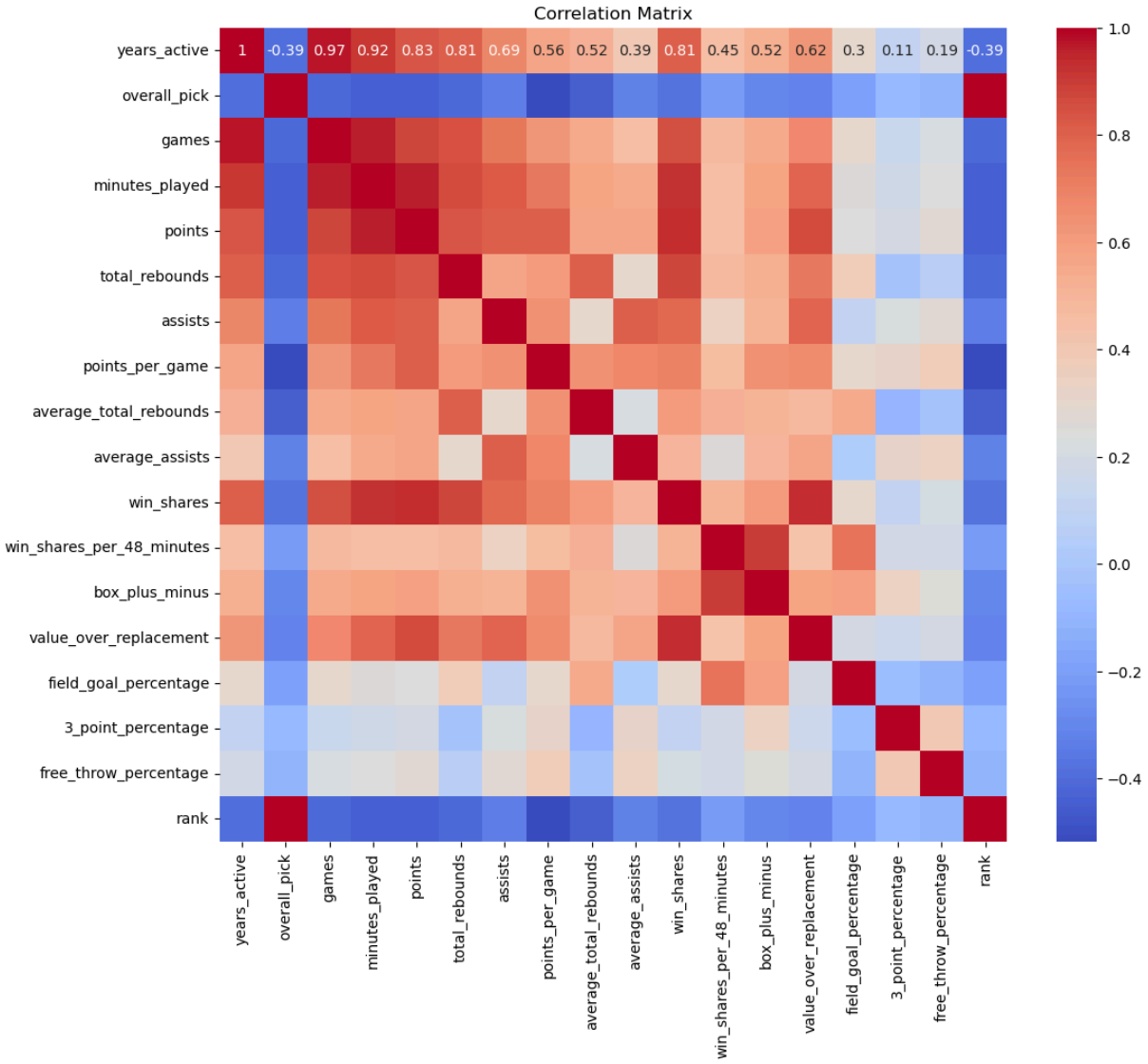
In [16]:
```
# Drop rows with missing values in the relevant columns
relevant_columns = ['years_active', 'overall_pick', 'games', 'minutes_played
                    'assists', 'points_per_game', 'average_total_rebounds',
                    'win_shares_per_48_minutes', 'box_plus_minus', 'value_ov
                    'field_goal_percentage', '3_point_percentage', 'free_thr
nba_draft_data = nba_draft_data.dropna(subset=relevant_columns)
```

Building and Evaluating a Linear Regression Model for NBA Player Career Longevity

In [17]:
```
# Select the independent and dependent variables
X = nba_draft_data[relevant_columns].drop(columns=['years_active'])
y = nba_draft_data['years_active']

# Add a constant to the model (intercept)
X = sm.add_constant(X)

# Fit the model
model = sm.OLS(y, X).fit()

# Print the summary
print(model.summary())

# Visualize the correlation matrix
plt.figure(figsize=(12, 10))
correlation_matrix = nba_draft_data[relevant_columns].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

```
                            OLS Regression Results
==============================================================================
==
Dep. Variable:           years_active   R-squared:                       0.9
55
Model:                            OLS   Adj. R-squared:                  0.9
54
Method:                 Least Squares   F-statistic:                     198
6.
Date:                Thu, 30 May 2024   Prob (F-statistic):               0.
00
Time:                        12:57:39   Log-Likelihood:                -215
2.9
No. Observations:                1529   AIC:                             434
0.
Df Residuals:                    1512   BIC:                             443
0.
Df Model:                          16
Covariance Type:            nonrobust
==============================================================================
=================
                                coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
-----------------
const                          2.4942      0.394      6.335      0.000
1.722       3.267
overall_pick                  -0.0011      0.001     -1.114      0.265      -
0.003       0.001
games                          0.0177      0.000     45.269      0.000
0.017       0.018
minutes_played                -0.0001   2.18e-05     -6.365      0.000      -
0.000   -9.59e-05
points                         0.0001   3.57e-05      2.870      0.004    3.2
5e-05       0.000
total_rebounds              1.586e-05   5.83e-05      0.272      0.786   -9.8
5e-05       0.000
assists                    -4.753e-08   8.02e-05     -0.001      1.000      -
0.000       0.000
points_per_game               -0.0314      0.020     -1.535      0.125      -
0.072       0.009
average_total_rebounds        -0.0133      0.037     -0.355      0.722      -
0.087       0.060
average_assists               -0.0622      0.050     -1.241      0.215      -
0.160       0.036
win_shares                    -0.0185      0.006     -2.861      0.004      -
0.031      -0.006
win_shares_per_48_minutes      0.5383      1.297      0.415      0.678      -
2.006       3.083
box_plus_minus                 0.0218      0.029      0.747      0.455      -
0.035       0.079
value_over_replacement         0.0245      0.011      2.187      0.029
0.003       0.046
field_goal_percentage         -0.5147      0.671     -0.767      0.443      -
1.830       0.801
3_point_percentage            -0.6889      0.252     -2.737      0.006      -
```

```
1.183      -0.195
free_throw_percentage         -0.6308       0.298      -2.116       0.035       -
1.216      -0.046
rank                          -0.0011       0.001      -1.114       0.265       -
0.003       0.001
==============================================================================
==
Omnibus:                      279.475   Durbin-Watson:                   1.8
70
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              664.2
91
Skew:                           1.003   Prob(JB):                     5.64e-1
45
Kurtosis:                       5.530   Cond. No.                      6.42e+
17
==============================================================================
==

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
[2] The smallest eigenvalue is 8.45e-25. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```



Calculating Variance Inflation Factors (VIF) to Detect Multicollinearity

In [18]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Calculate VIF for each predictor
X = nba_draft_data[relevant_columns].drop(columns=['years_active'])
X = sm.add_constant(X)
vif_data = pd.DataFrame()
vif_data["feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(ler

print(vif_data)
```

	feature	VIF
0	const	239.565719
1	overall_pick	inf
2	games	24.961591
3	minutes_played	73.011214
4	points	47.829147
5	total_rebounds	22.013521
6	assists	17.327856
7	points_per_game	15.615536
8	average_total_rebounds	9.454397
9	average_assists	8.700687
10	win_shares	53.480638
11	win_shares_per_48_minutes	13.208691
12	box_plus_minus	13.930842
13	value_over_replacement	27.406733
14	field_goal_percentage	3.222193
15	3_point_percentage	1.589456
16	free_throw_percentage	1.620110
17	rank	inf

```
/opt/anaconda3/lib/python3.11/site-packages/statsmodels/stats/outliers_influ
ence.py:198: RuntimeWarning: divide by zero encountered in scalar divide
  vif = 1. / (1. - r_squared_i)
```

Cleaning Data and Fitting the OLS Regression Model

In [19]:
```
# Remove rows with missing values in the selected columns
selected_columns = ['years_active', 'overall_pick', 'games', 'points_per_gam
                    'box_plus_minus', 'value_over_replacement', 'field_goal_
                    'free_throw_percentage']
nba_draft_data_reduced = nba_draft_data.dropna(subset=selected_columns)

# Select the independent and dependent variables
X = nba_draft_data_reduced[selected_columns].drop(columns=['years_active'])
y = nba_draft_data_reduced['years_active']
```

```
# Add a constant to the model (intercept)
X = sm.add_constant(X)

# Fit the model
model = sm.OLS(y, X).fit()

# Print the summary
print(model.summary())

# Recalculate VIF for the selected columns
vif_data_reduced = pd.DataFrame()
vif_data_reduced["feature"] = X.columns
vif_data_reduced["VIF"] = [variance_inflation_factor(X.values, i) for i in r

print(vif_data_reduced)
```

```
                            OLS Regression Results
==============================================================================
==
Dep. Variable:           years_active   R-squared:                       0.9
52
Model:                            OLS   Adj. R-squared:                  0.9
52
Method:                 Least Squares   F-statistic:                     303
4.
Date:                Thu, 30 May 2024   Prob (F-statistic):               0.
00
Time:                        12:57:39   Log-Likelihood:                -218
9.7
No. Observations:                1529   AIC:                             440
1.
Df Residuals:                    1518   BIC:                             446
0.
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
==============
                             coef    std err          t      P>|t|      [0.0
25      0.975]
------------------------------------------------------------------------------
--------------
const                      2.7561      0.385      7.161      0.000       2.0
01       3.511
overall_pick              -0.0026      0.002     -1.330      0.184      -0.0
07       0.001
games                      0.0147      0.000    122.394      0.000       0.0
14       0.015
points_per_game           -0.0326      0.012     -2.801      0.005      -0.0
55      -0.010
average_total_rebounds    -0.0564      0.022     -2.566      0.010      -0.0
99      -0.013
average_assists           -0.0984      0.027     -3.608      0.000      -0.1
52      -0.045
box_plus_minus             0.0483      0.014      3.329      0.001       0.0
20       0.077
value_over_replacement    -0.0117      0.003     -3.354      0.001      -0.0
19      -0.005
field_goal_percentage     -0.1742      0.576     -0.303      0.762      -1.3
03       0.955
3_point_percentage        -0.7672      0.254     -3.021      0.003      -1.2
65      -0.269
free_throw_percentage     -0.5801      0.289     -2.008      0.045      -1.1
47      -0.013
==============================================================================
==
Omnibus:                      264.434   Durbin-Watson:                   1.8
38
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              632.4
84
Skew:                           0.951   Prob(JB):                     4.55e-1
38
Kurtosis:                       5.512   Cond. No.                      1.28e+
```

```
04
================================================================================
==

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
[2] The condition number is large, 1.28e+04. This might indicate that there
are
strong multicollinearity or other numerical problems.
                   feature         VIF
0                    const  219.047337
1              overall_pick    1.462082
2                     games    2.243851
3           points_per_game    4.836569
4   average_total_rebounds    3.114978
5           average_assists    2.465082
6            box_plus_minus    3.281686
7   value_over_replacement    2.551091
8    field_goal_percentage    2.270520
9       3_point_percentage    1.548084
10   free_throw_percentage    1.455424
```

Standardizing Data, Performing PCA, and Fitting an OLS Regression Model

In [20]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Remove rows with missing values in the selected columns
selected_columns = ['years_active', 'games', 'points_per_game', 'average_tot
                    'box_plus_minus', 'value_over_replacement', 'field_goal_
                    'free_throw_percentage']
nba_draft_data_reduced = nba_draft_data.dropna(subset=selected_columns)

# Select the independent and dependent variables
X = nba_draft_data_reduced[selected_columns].drop(columns=['years_active'])
y = nba_draft_data_reduced['years_active']

# Standardize the predictors
X_standardized = StandardScaler().fit_transform(X)

# Perform PCA
pca = PCA(n_components=0.95)  # Retain 95% of the variance
X_pca = pca.fit_transform(X_standardized)

# Create a DataFrame for the principal components
X_pca_df = pd.DataFrame(X_pca, columns=[f'PC{i+1}' for i in range(X_pca.shap

# Reset index to ensure alignment
X_pca_df.reset_index(drop=True, inplace=True)
y.reset_index(drop=True, inplace=True)
```

```
# Add the constant term back
X_pca_df = sm.add_constant(X_pca_df)

# Fit the regression model using the principal components
model_pca = sm.OLS(y, X_pca_df).fit()

# Print the summary
print(model_pca.summary())
```

```
                            OLS Regression Results
==============================================================================
==
Dep. Variable:           years_active   R-squared:                       0.9
51
Model:                            OLS   Adj. R-squared:                  0.9
50
Method:                 Least Squares   F-statistic:                     418
3.
Date:                Thu, 30 May 2024   Prob (F-statistic):               0.
00
Time:                        12:57:40   Log-Likelihood:                -221
6.9
No. Observations:                1529   AIC:                             445
0.
Df Residuals:                    1521   BIC:                             449
3.
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
const          6.7286      0.026    254.407      0.000       6.677       6.7
80
PC1            1.6707      0.013    128.944      0.000       1.645       1.6
96
PC2            0.4400      0.020     21.718      0.000       0.400       0.4
80
PC3           -1.0999      0.028    -39.628      0.000      -1.154      -1.0
45
PC4           -0.4997      0.033    -15.060      0.000      -0.565      -0.4
35
PC5           -2.6697      0.037    -71.438      0.000      -2.743      -2.5
96
PC6            1.3819      0.040     34.596      0.000       1.304       1.4
60
PC7           -3.0356      0.048    -63.892      0.000      -3.129      -2.9
42
==============================================================================
==
Omnibus:                      229.760   Durbin-Watson:                   1.8
24
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              484.0
24
Skew:                           0.880   Prob(JB):                     7.86e-1
06
Kurtosis:                       5.121   Cond. No.                         3.
67
==============================================================================
==

Notes:
```

```
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
```

Displaying PCA components

In [21]:
```
# Get the PCA components
pca_components = pd.DataFrame(pca.components_, columns=X.columns, index=[f'P

# Display the PCA components
print(pca_components)
```

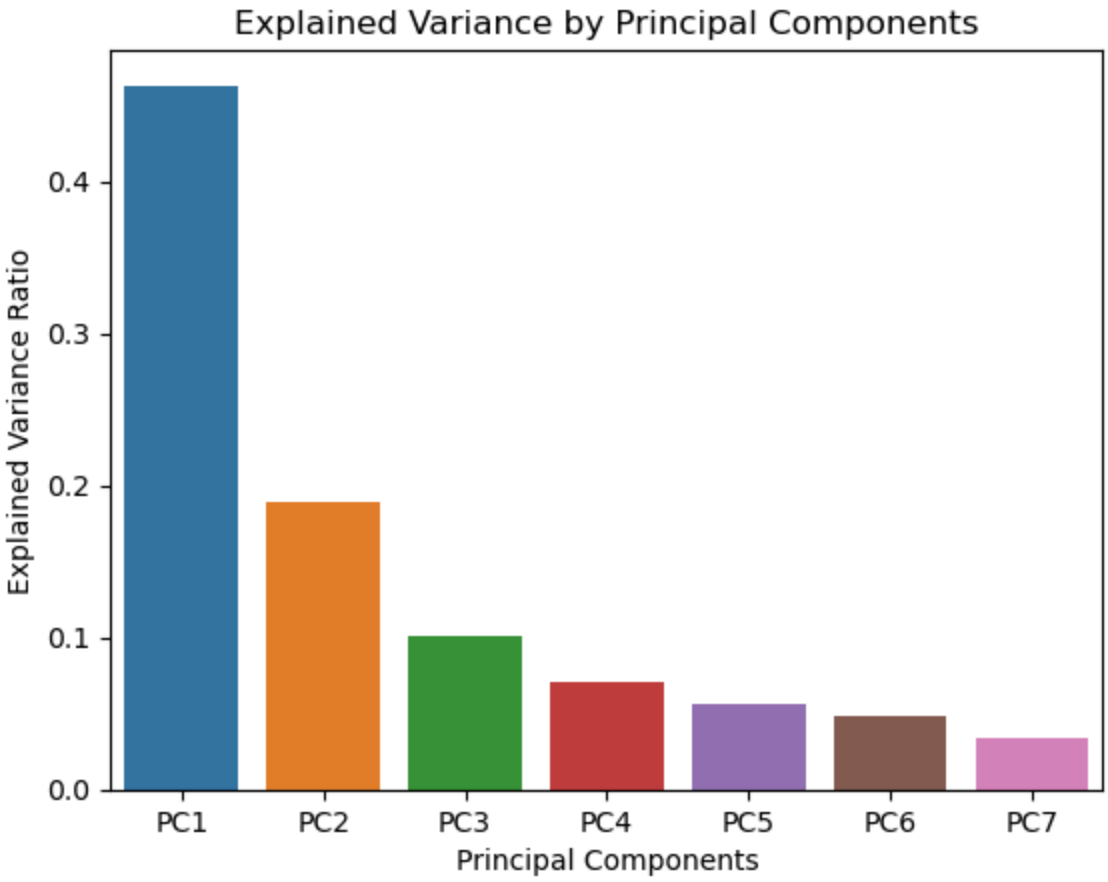
```
       games  points_per_game  average_total_rebounds  average_assists  \
PC1  0.386238         0.437626                0.334559         0.341358
PC2  0.072599        -0.065040                0.411774        -0.293943
PC3 -0.251097        -0.093114               -0.059922        -0.248591
PC4 -0.105222        -0.072309               -0.253411         0.334530
PC5 -0.552282         0.211512               -0.045451         0.609108
PC6  0.258054        -0.436128               -0.593331         0.070599
PC7 -0.628920         0.007911                0.078606        -0.324570

     box_plus_minus  value_over_replacement  field_goal_percentage  \
PC1        0.406956                0.390869               0.228907
PC2        0.068405               -0.008463               0.484021
PC3        0.367293               -0.371526               0.550908
PC4        0.152854                0.182802              -0.033230
PC5        0.086808               -0.251078               0.180114
PC6        0.330277                0.258394               0.238315
PC7        0.189319                0.659061              -0.107443

     3_point_percentage  free_throw_percentage
PC1            0.164989               0.181210
PC2           -0.497420              -0.498081
PC3            0.513036               0.152484
PC4            0.360488              -0.787233
PC5           -0.406944               0.092461
PC6           -0.347177               0.181654
PC7           -0.027614               0.101798
```

Visualizing PCA Results, Cross-Validation, and Residual Analysis

In [22]:
```
# Plot the explained variance ratio
explained_variance = pca.explained_variance_ratio_
sns.barplot(x=[f'PC{i+1}' for i in range(len(explained_variance))], y=explai
plt.xlabel('Principal Components')
plt.ylabel('Explained Variance Ratio')
plt.title('Explained Variance by Principal Components')
plt.show()

# Cross-validation
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
```

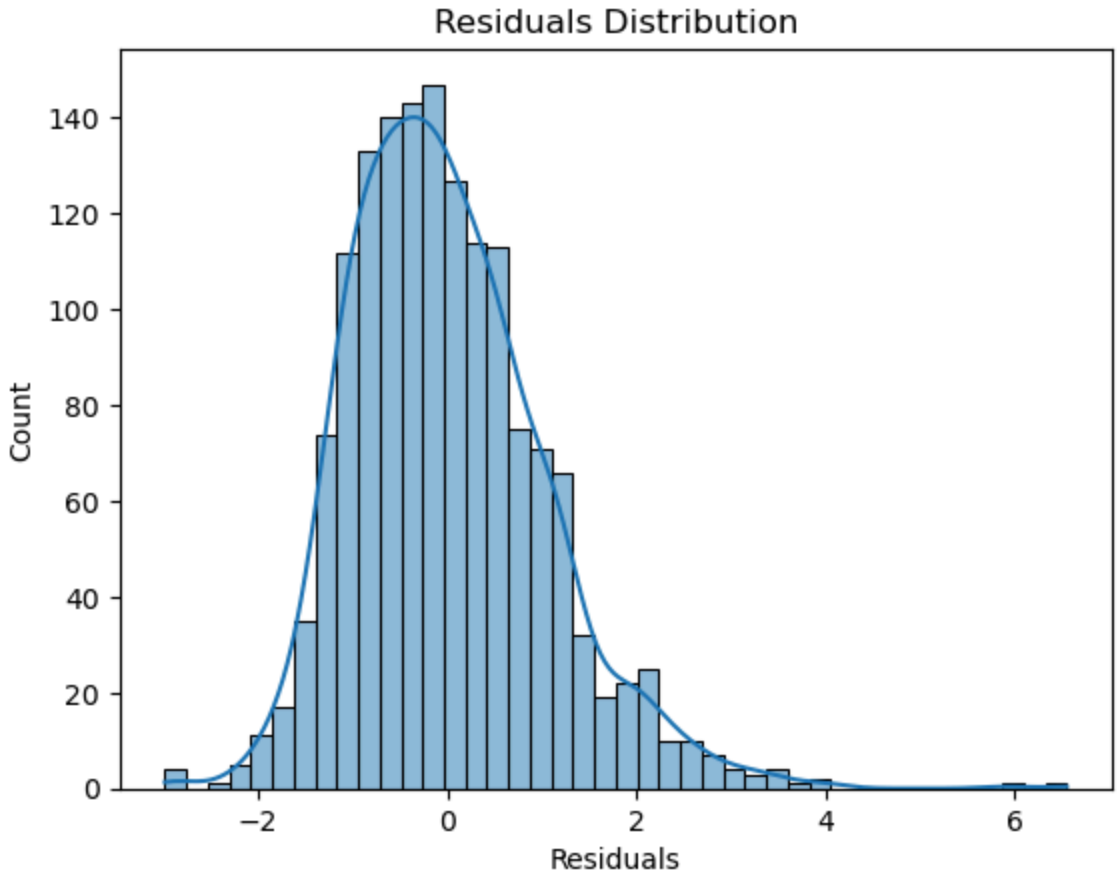
```
# Cross-validate the PCA model
model = LinearRegression()
scores = cross_val_score(model, X_pca, y, cv=5)
print(f'Cross-validated R-squared: {scores.mean()}')

# Residual analysis
residuals = model_pca.resid
sns.histplot(residuals, kde=True)
plt.xlabel('Residuals')
plt.title('Residuals Distribution')
plt.show()

sns.scatterplot(x=model_pca.fittedvalues, y=residuals)
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.title('Residuals vs Fitted Values')
plt.axhline(0, color='red', linestyle='--')
plt.show()
```

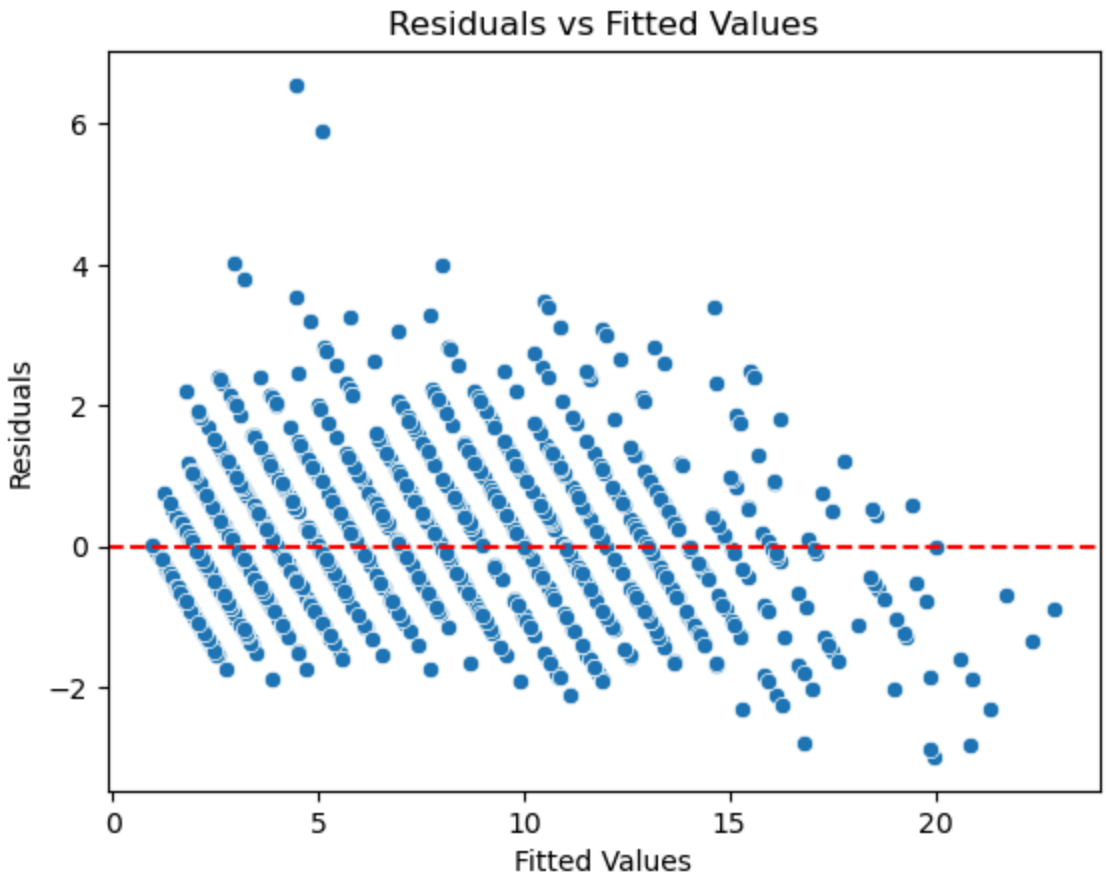
```
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1765: Future
Warning: unique with argument that is not not a Series, Index, ExtensionArra
y, or np.ndarray is deprecated and will raise in a future version.
  order = pd.unique(vector)
```




```
Cross-validated R-squared: 0.860028715167446
```

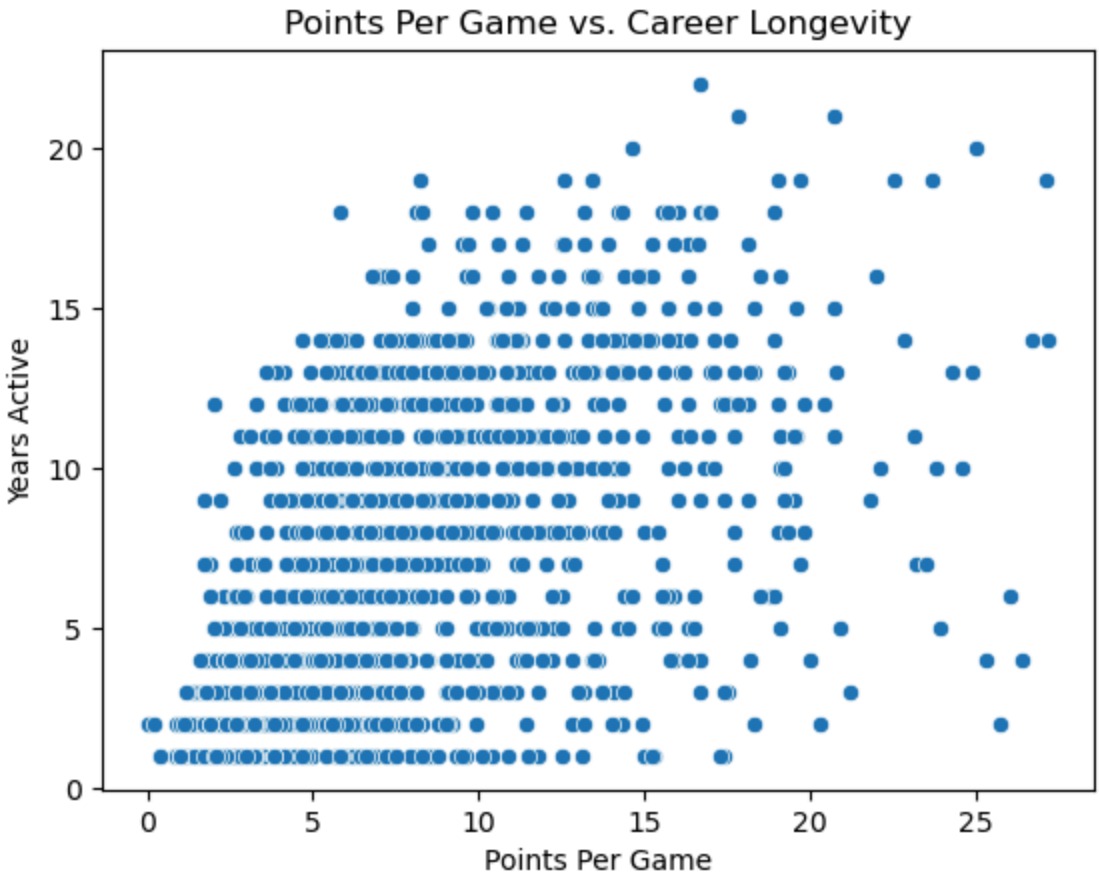
```
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

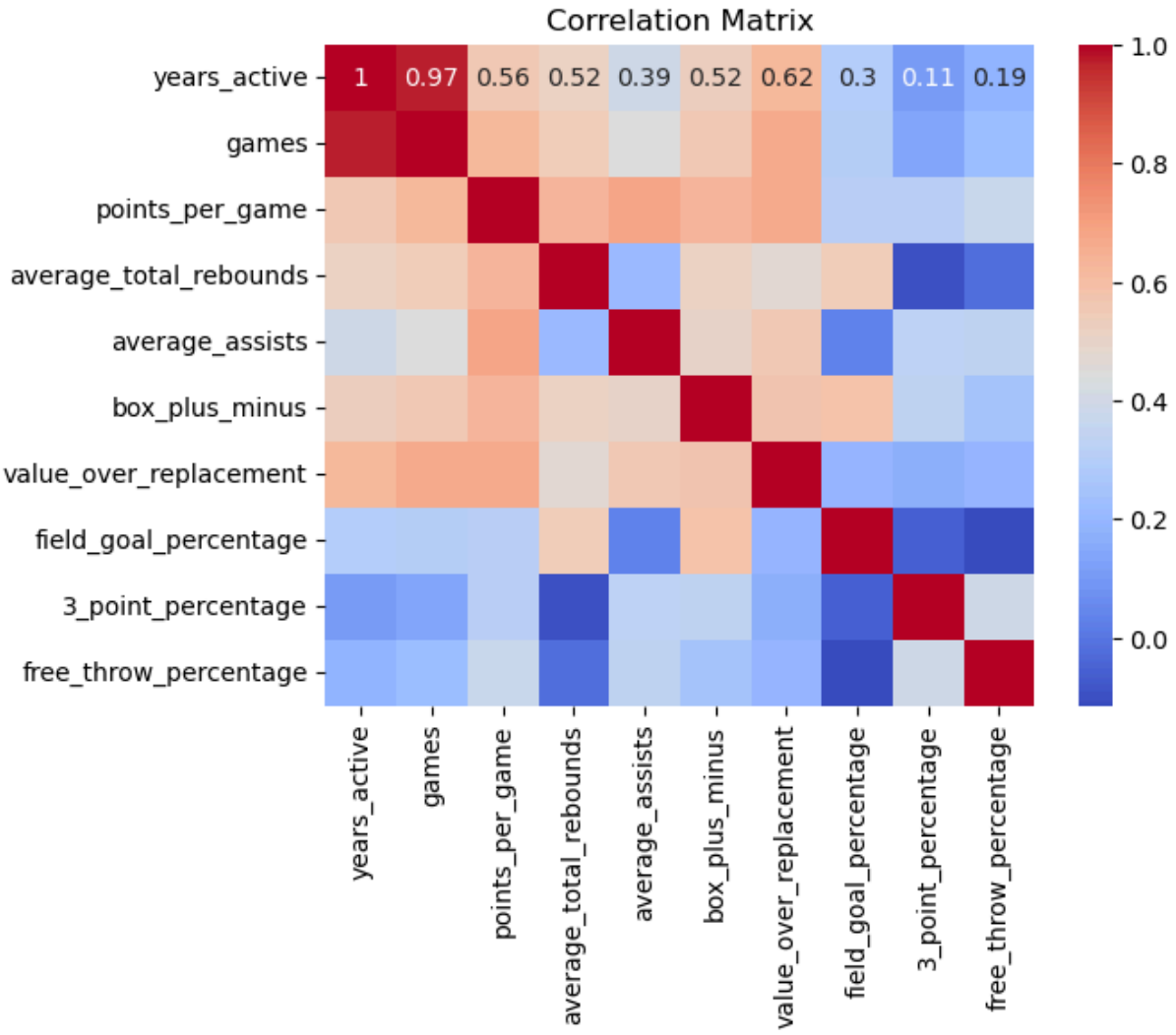

Residuals Distribution
Count
Residuals
0
20
40
60
80
100
120
140
−2
0
2
4
6


Residuals vs Fitted Values
Residuals
Fitted Values
6
4
2
0
−2
0
5
10
15
20

Scatter plot for points per game vs. years active

In [23]:
```
# Scatter plot for points per game vs. years active
sns.scatterplot(data=nba_draft_data, x='points_per_game', y='years_active')
plt.title('Points Per Game vs. Career Longevity')
plt.xlabel('Points Per Game')
plt.ylabel('Years Active')
plt.show()

# Heatmap for correlation matrix
corr_matrix = nba_draft_data[['years_active', 'games', 'points_per_game',
                              'average_total_rebounds', 'average_assists',
                              'box_plus_minus', 'value_over_replacement',
                              'field_goal_percentage', '3_point_percentage',
                              'free_throw_percentage']].corr()

sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```





Linear Regression Model

In [24]:
```
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Split data into training and test sets
X = nba_draft_data[['games', 'points_per_game', 'average_total_rebounds',
                    'average_assists', 'box_plus_minus', 'value_over_replace
                    'field_goal_percentage', '3_point_percentage', 'free_thr
y = nba_draft_data['years_active']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran

# Standardize the predictors
scaler = StandardScaler().fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Perform PCA
pca = PCA(n_components=0.95)
```

```
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)

# Fit linear regression model
model = LinearRegression()
model.fit(X_train_pca, y_train)

# Predict on test set
y_pred = model.predict(X_test_pca)

# Evaluate model
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')

# Cross-validation
cv_scores = cross_val_score(model, X_train_pca, y_train, cv=5, scoring='neg_
print(f'Cross-validated MSE: {-cv_scores.mean()}')
```

```
Mean Squared Error: 0.9457133683347967
Cross-validated MSE: 1.114889217513488
```

In [25]:
```
# Fit model to entire dataset
model.fit(X_pca_df, y)

# Extract coefficients
coefficients = model.coef_

# Create a DataFrame to display coefficients
importance_df = pd.DataFrame({
    'Principal Component': [f'PC{i+1}' for i in range(len(coefficients))],
    'Coefficient': coefficients
})

print(importance_df.sort_values(by='Coefficient', ascending=False))
```

```
  Principal Component  Coefficient
1                 PC2     1.670716
6                 PC7     1.381856
2                 PC3     0.439962
0                 PC1     0.000000
4                 PC5    -0.499669
3                 PC4    -1.099868
5                 PC6    -2.669683
7                 PC8    -3.035632
```

Scenario Analysis for Career Longevity Prediction

In [26]:
```
# Create a DataFrame for scenario analysis
scenarios = pd.DataFrame({
    'games': [50, 70, 90],
    'points_per_game': [10, 15, 20],
    'average_total_rebounds': [5, 7, 10],
    'average_assists': [2, 5, 7],
```

```
    'box_plus_minus': [0, 5, 10],
    'value_over_replacement': [0.5, 1, 1.5],
    'field_goal_percentage': [0.45, 0.5, 0.55],
    '3_point_percentage': [0.35, 0.4, 0.45],
    'free_throw_percentage': [0.75, 0.8, 0.85]
})

# Standardize the scenario data
scenarios_scaled = scaler.transform(scenarios)

# Apply PCA transformation
scenarios_pca = pca.transform(scenarios_scaled)

# Predict career longevity for each scenario
scenarios_pca_df = pd.DataFrame(scenarios_pca, columns=[f'PC{i+1}' for i in
scenarios_pca_df = sm.add_constant(scenarios_pca_df)
scenarios['predicted_years_active'] = model_pca.predict(scenarios_pca_df)

print(scenarios)
```

```
   games  points_per_game  average_total_rebounds  average_assists  \
0     50               10                       5                2
1     70               15                       7                5
2     90               20                      10                7

   box_plus_minus  value_over_replacement  field_goal_percentage  \
0               0                     0.5                   0.45
1               5                     1.0                   0.50
2              10                     1.5                   0.55

   3_point_percentage  free_throw_percentage  predicted_years_active
0                0.35                   0.75                1.778380
1                0.40                   0.80                1.970082
2                0.45                   0.85                2.072749
```

Data Preparation for Clustering Analysis

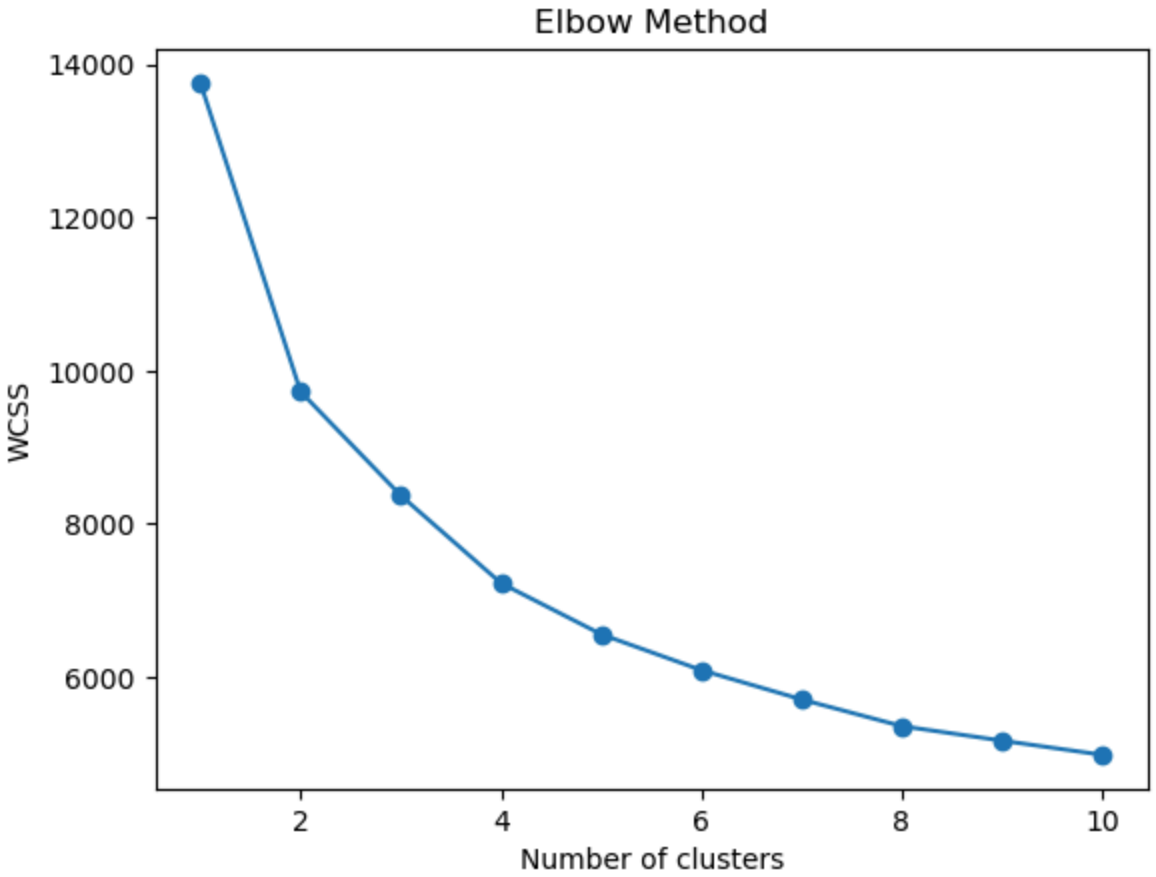
In [27]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

#Select the relevant features for clustering
features = ['games', 'points_per_game', 'average_total_rebounds', 'average_a
            'box_plus_minus', 'value_over_replacement', 'field_goal_percenta
            '3_point_percentage', 'free_throw_percentage']

X = nba_draft_data[features]

# Handle missing values by imputing the mean
```

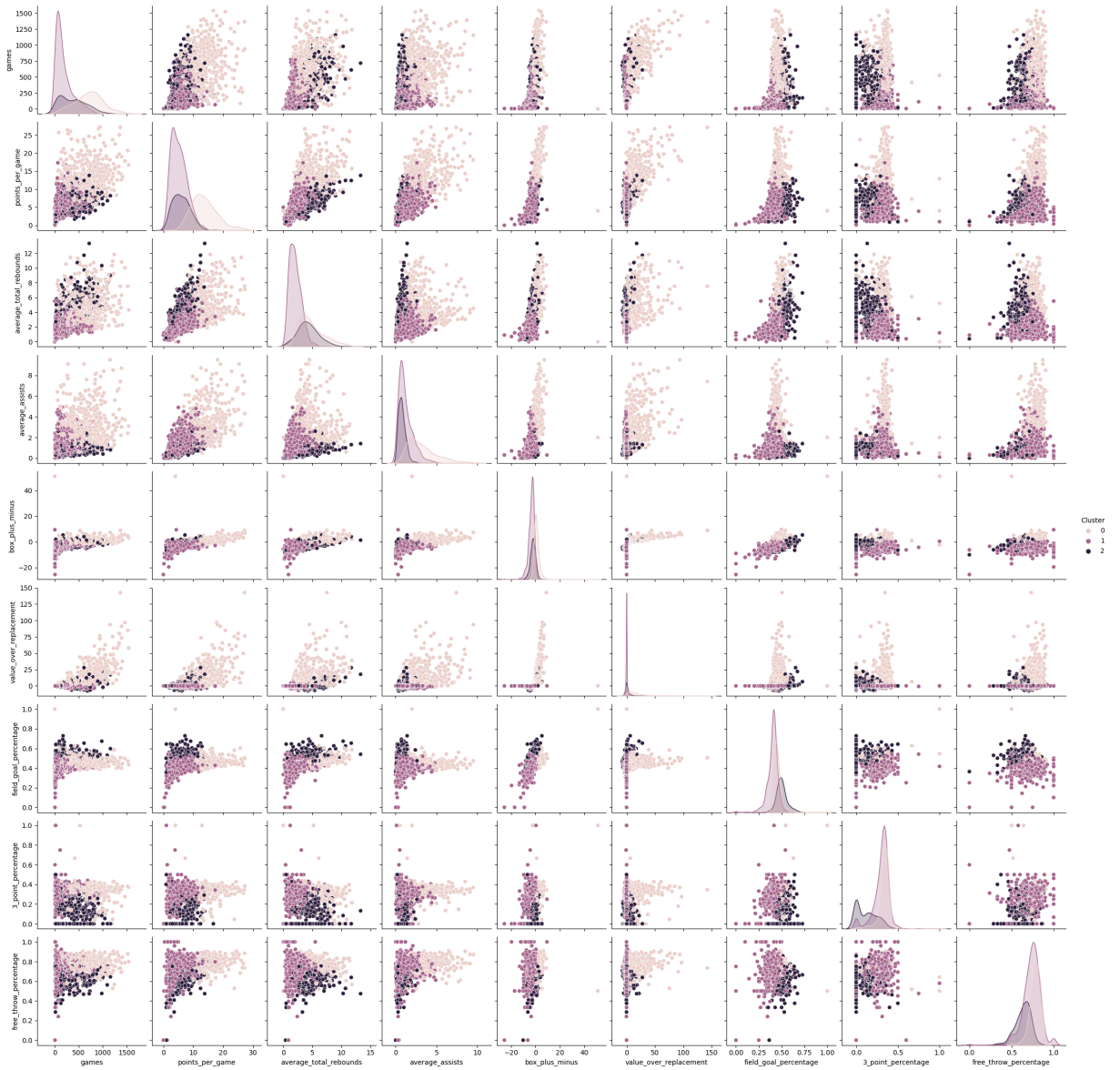
```
imputer = SimpleImputer(strategy='mean')
X_imputed = imputer.fit_transform(X)  # First fit, then transform

# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_imputed)
```

Clustering Analysis

In [28]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
# Use the Elbow Method to find the optimal number of clusters
wcss = []  # Within-cluster sum of squares
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=42, n_init=3)  # Set n_init e
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)

# Plot the Elbow Method
plt.plot(range(1, 11), wcss, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

# Based on the Elbow Method, choose an appropriate number of clusters
kmeans = KMeans(n_clusters=3, random_state=42, n_init=3)  # Set n_init expli
clusters = kmeans.fit_predict(X_scaled)

# Add cluster labels to the original data
nba_draft_data['Cluster'] = clusters

# Visualize clusters
sns.pairplot(nba_draft_data, hue='Cluster', vars=features)
plt.show()

# Summarize cluster characteristics, selecting only numeric columns
numeric_columns = nba_draft_data.select_dtypes(include=['number']).columns
cluster_summary = nba_draft_data.groupby('Cluster')[numeric_columns].mean()
print(cluster_summary)
```


Elbow Method
14000
12000
10000
8000
6000
WCSS
2
4
6
8
10
Number of clusters

```
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```


games
points_per_game
average_total_rebounds
average_assists
box_plus_minus
value_over_replacement
field_goal_percentage
3_point_percentage
free_throw_percentage
Cluster
0
1
2

```
                  id         year       rank  overall_pick  years_active  \
Cluster
0         895.244186  2004.430233  16.137209     16.137209     11.123256
1        1067.408322  2007.085826  31.016905     31.016905      4.144343
2         858.378788  2003.609091  26.212121     26.212121      7.024242

              games  minutes_played       points  total_rebounds      assist
s  \
Cluster
0        710.846512    20727.609302  9664.900000     3468.244186  2267.16511
6
1        187.919376     3320.695709  1211.101430      450.392718   270.03641
1
2        374.190909     7438.466667  2595.618182     1925.230303   324.22424
2

         ...  free_throw_percentage  average_minutes_played  points_per_game
\
Cluster  ...
0        ...               0.781119               28.867907        13.603721
1        ...               0.735078               14.577633         5.204291
2        ...               0.632618               16.830606         5.967879

         average_total_rebounds  average_assists  win_shares  \
Cluster
0                      4.832558         3.193953   49.700930
1                      2.046424         1.205202    3.928479
2                      4.317879         0.753939   15.952727

         win_shares_per_48_minutes  box_plus_minus  value_over_replacement
\
Cluster
0                         0.111572        0.940465               16.462326
1                         0.031394       -3.460728               -0.120286
2                         0.090612       -2.157273                1.153030

         Cluster
Cluster
0            0.0
1            1.0
2            2.0

[3 rows x 22 columns]
```

Random Forest Classifier for Predicting Player Clusters

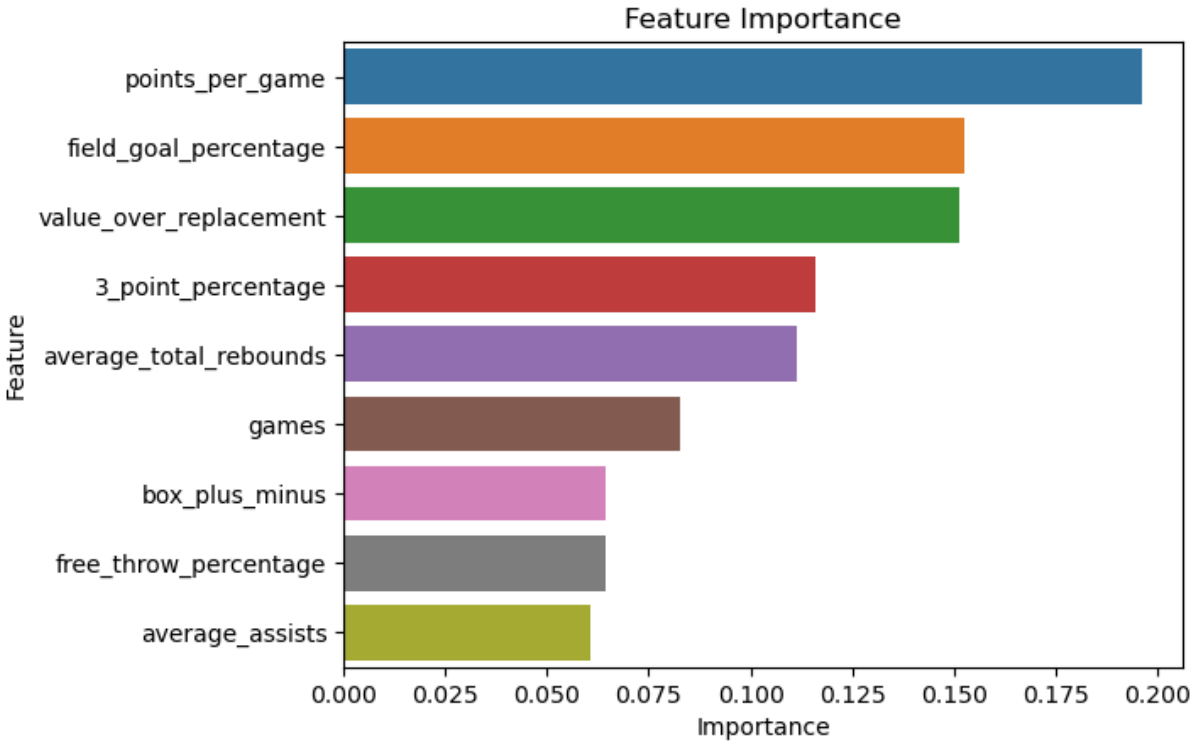
In [29]:
```
# Define the features and the target variable
X = nba_draft_data[features]
y = nba_draft_data['Cluster']  # Ensure the 'Cluster' column is already crea

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rar
```

```
# Train a Random Forest Classifier
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)

# Predict on test set
y_pred = clf.predict(X_test)

# Evaluate the model
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

# Feature Importance
feature_importances = pd.DataFrame(clf.feature_importances_, index=features,
print(feature_importances)
```

```
[[ 87   2   0]
 [  0 138   2]
 [  3  11  63]]
              precision    recall  f1-score   support

           0       0.97      0.98      0.97        89
           1       0.91      0.99      0.95       140
           2       0.97      0.82      0.89        77

    accuracy                           0.94       306
   macro avg       0.95      0.93      0.94       306
weighted avg       0.94      0.94      0.94       306

                        importance
points_per_game           0.196383
field_goal_percentage     0.152552
value_over_replacement    0.151438
3_point_percentage        0.115946
average_total_rebounds    0.111296
games                     0.082795
box_plus_minus            0.064631
free_throw_percentage     0.064454
average_assists           0.060504
```

Plotting Feature Importance for Random Forest Classifier

In [30]:
```
# Plot feature importance
sns.barplot(x=feature_importances.importance, y=feature_importances.index)
plt.title('Feature Importance')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.show()
```



Clustering Players. Added in player names, draft year and their team

In [31]:
```
import numpy as np
# Include player names, draft year, and team
features_with_info = ['player', 'year', 'team', 'games', 'points_per_game',
                      'average_total_rebounds', 'average_assists', 'box_plus
                      'value_over_replacement', 'field_goal_percentage',
                      '3_point_percentage', 'free_throw_percentage']

X_info = nba_draft_data[features_with_info].copy()

# Convert infinite values to NaN
X_info.replace([np.inf, -np.inf], np.nan, inplace=True)

# Drop rows with NaN values
X_info.dropna(inplace=True)

# Select features for clustering
features = ['games', 'points_per_game', 'average_total_rebounds', 'average_a
            'box_plus_minus', 'value_over_replacement', 'field_goal_percenta
            '3_point_percentage', 'free_throw_percentage']

X = X_info[features]

# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Perform clustering
```

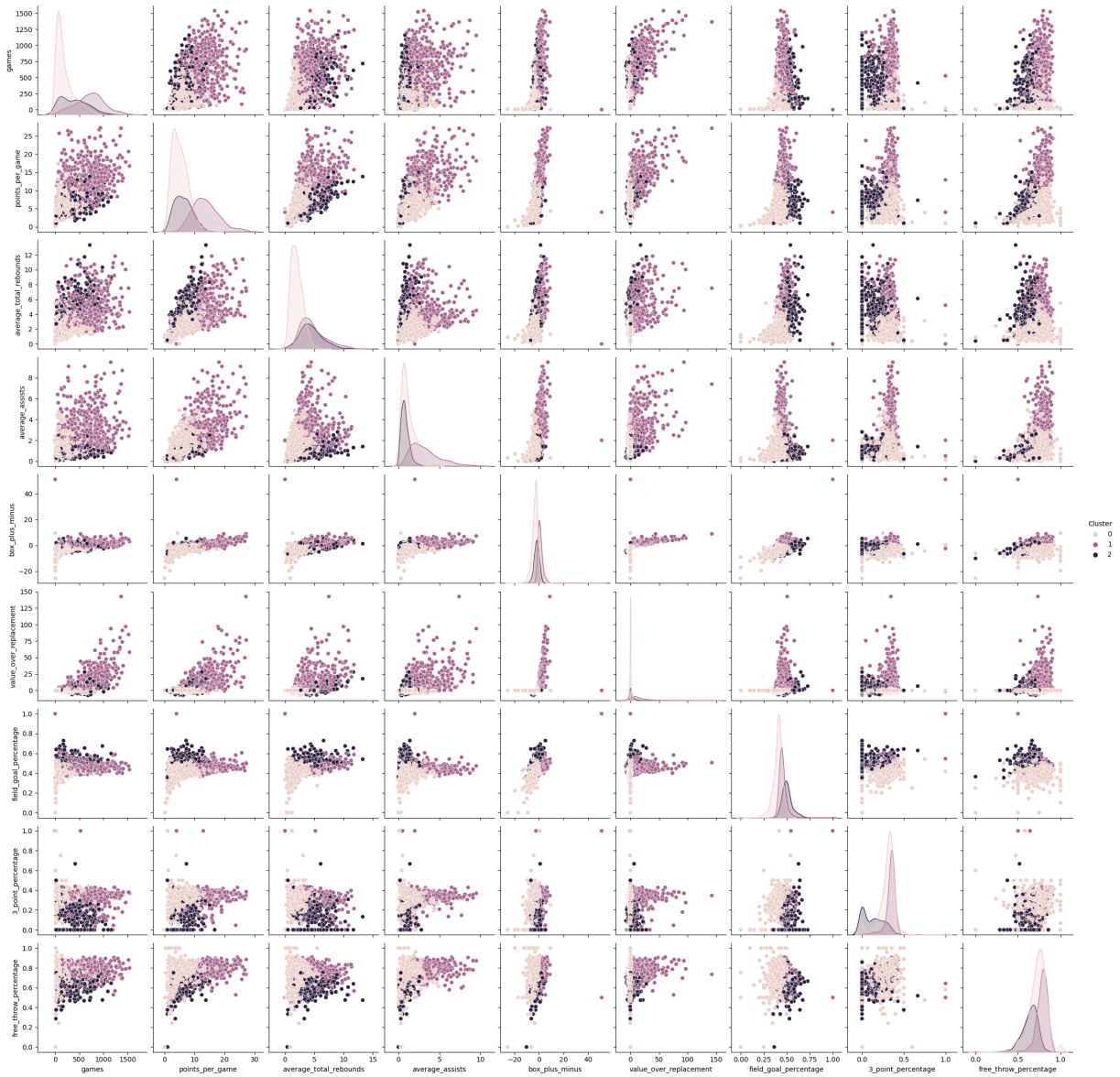
```
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
clusters = kmeans.fit_predict(X_scaled)

# Add cluster labels to the dataset
X_info['Cluster'] = clusters

# Summarize cluster characteristics (numeric features only)
numeric_columns = X_info.select_dtypes(include=[np.number]).columns
cluster_summary = X_info.groupby('Cluster')[numeric_columns].mean()
print(cluster_summary)
```

```
                year       games  points_per_game  average_total_rebounds  \
Cluster
0        2007.001289  187.511598         5.173454                2.038015
1        2004.289157  718.491566        13.700241                4.750361
2        2003.940828  384.538462         6.274852                4.507988

         average_assists  box_plus_minus  value_over_replacement  \
Cluster
0               1.200644       -3.488273               -0.117010
1               3.240723        0.969398               16.810602
2               0.805917       -1.965089                1.423669

         field_goal_percentage  3_point_percentage  free_throw_percentage  \
Cluster
0                     0.400567            0.298674               0.733272
1                     0.453402            0.343094               0.784108
2                     0.502497            0.130858               0.637562

         Cluster
Cluster
0            0.0
1            1.0
2            2.0
```

Plot Pair Plot with Player Information

In [32]:
```
sns.pairplot(X_info, hue='Cluster', vars=features)
plt.show()
```

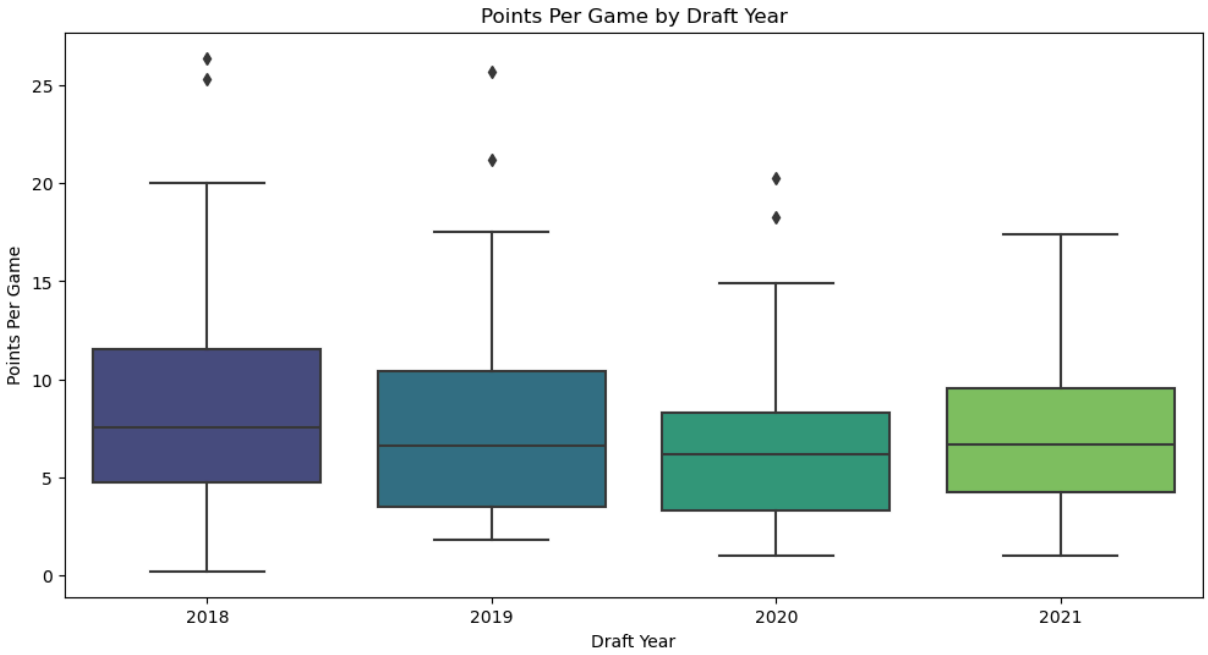
```
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Future
Warning: use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

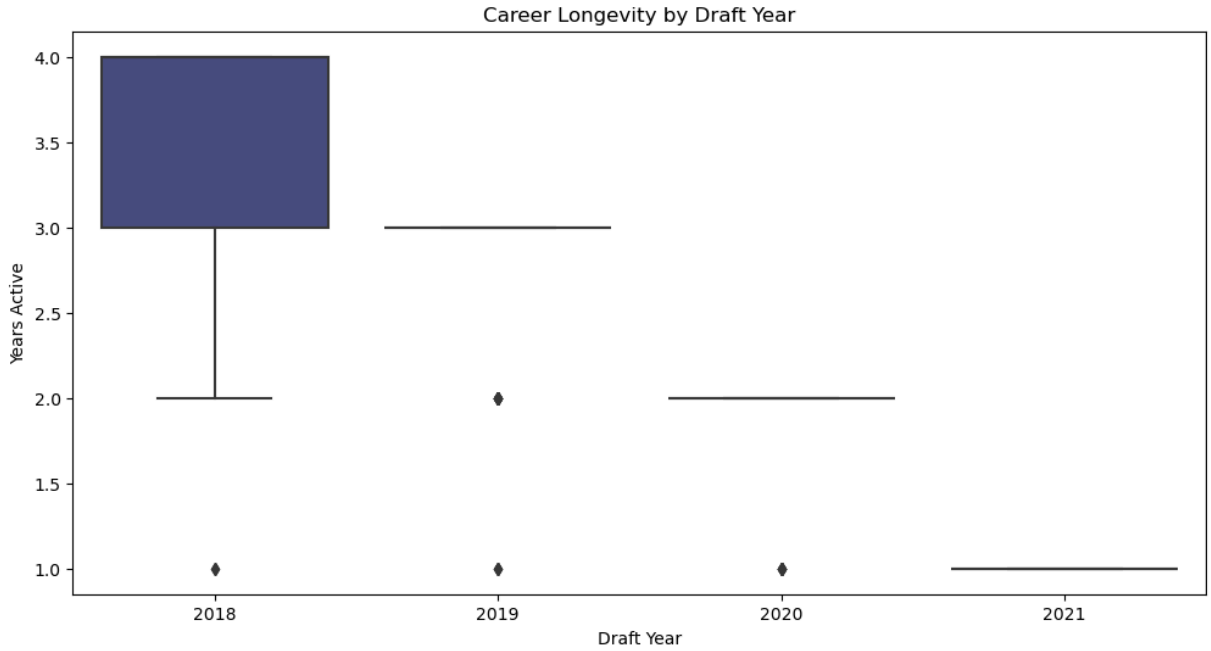


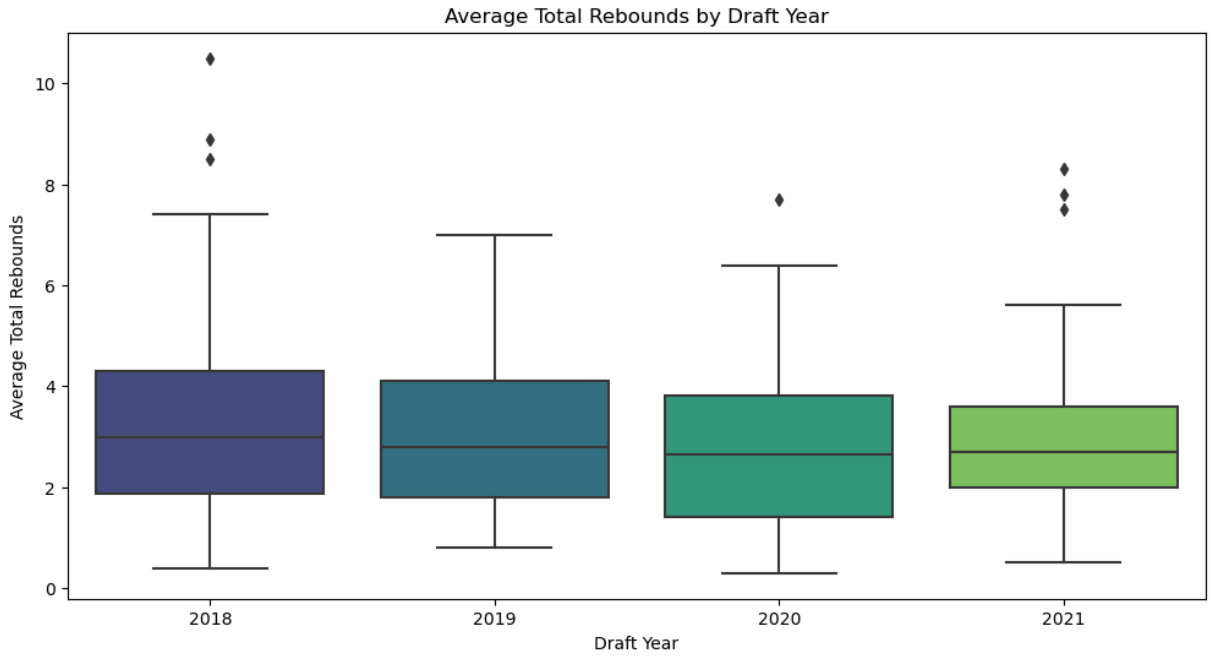
Visualize performance trends for draft classes between 2018-2022

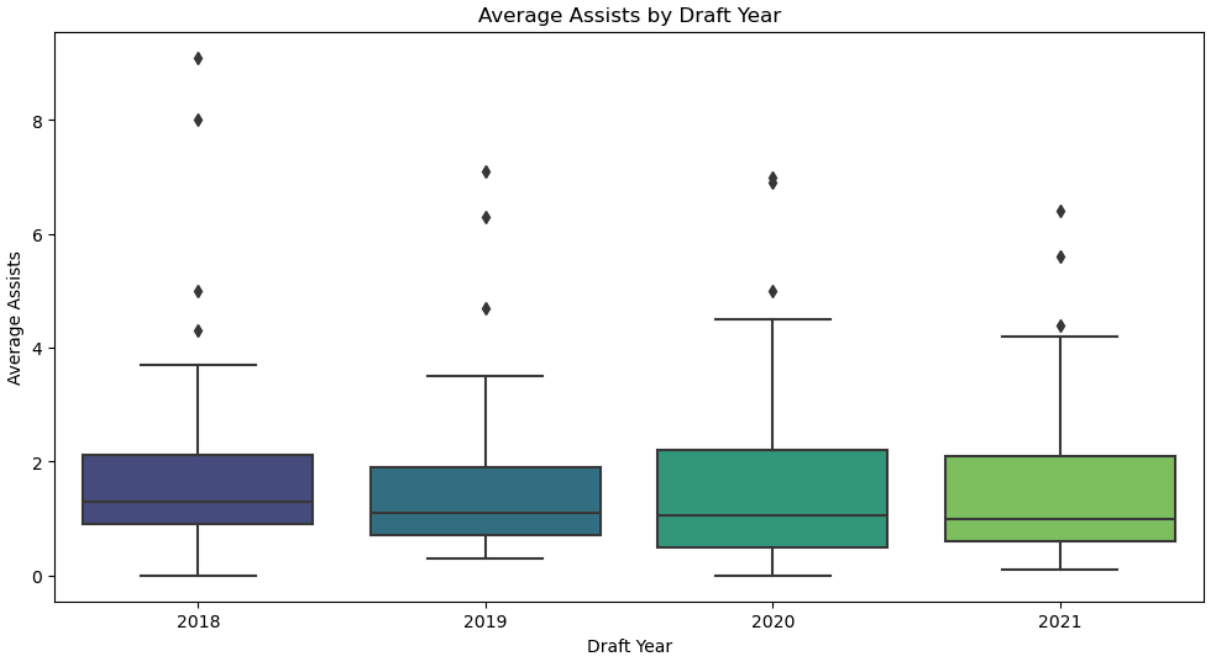
In [33]:
```
# Filter for recent draft classes (e.g., 2018-2022)
recent_years = nba_draft_data[nba_draft_data['year'] >= 2018]

# Ensure the 'Cluster' column exists and is not causing issues
if 'Cluster' not in recent_years.columns:
    # If the column doesn't exist, assign a default value or cluster the dat
    recent_years['Cluster'] = 0

# Function to visualize performance trends for different draft classes
def visualize_performance_trends(data, metric, title, xlabel, ylabel):
    plt.figure(figsize=(12, 6))
    sns.boxplot(data=data, x='year', y=metric, palette='viridis')
    plt.title(title)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
```

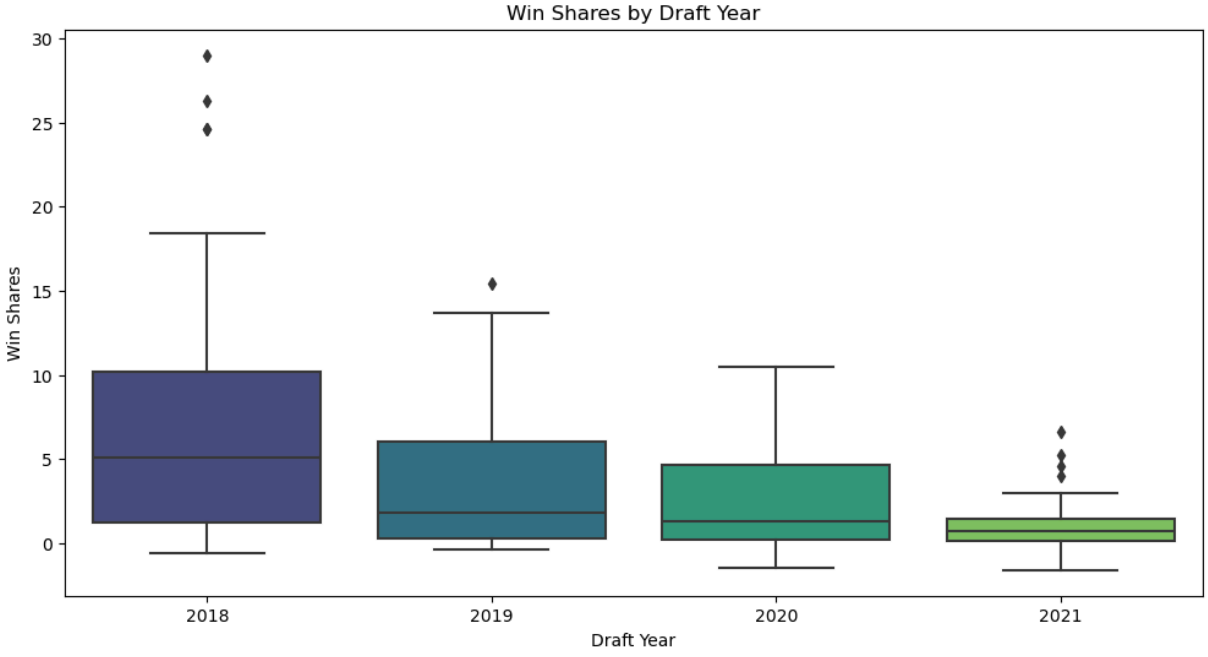
```
    plt.show()

# Points Per Game comparison
visualize_performance_trends(recent_years, 'points_per_game', 'Points Per Ga

# Career Longevity comparison
visualize_performance_trends(recent_years, 'years_active', 'Career Longevity

# Average Total Rebounds comparison
visualize_performance_trends(recent_years, 'average_total_rebounds', 'Averag

# Average Assists comparison
visualize_performance_trends(recent_years, 'average_assists', 'Average Assis

# Win Shares comparison
visualize_performance_trends(recent_years, 'win_shares', 'Win Shares by Draf

# Box Plus Minus comparison
visualize_performance_trends(recent_years, 'box_plus_minus', 'Box Plus Minus
```

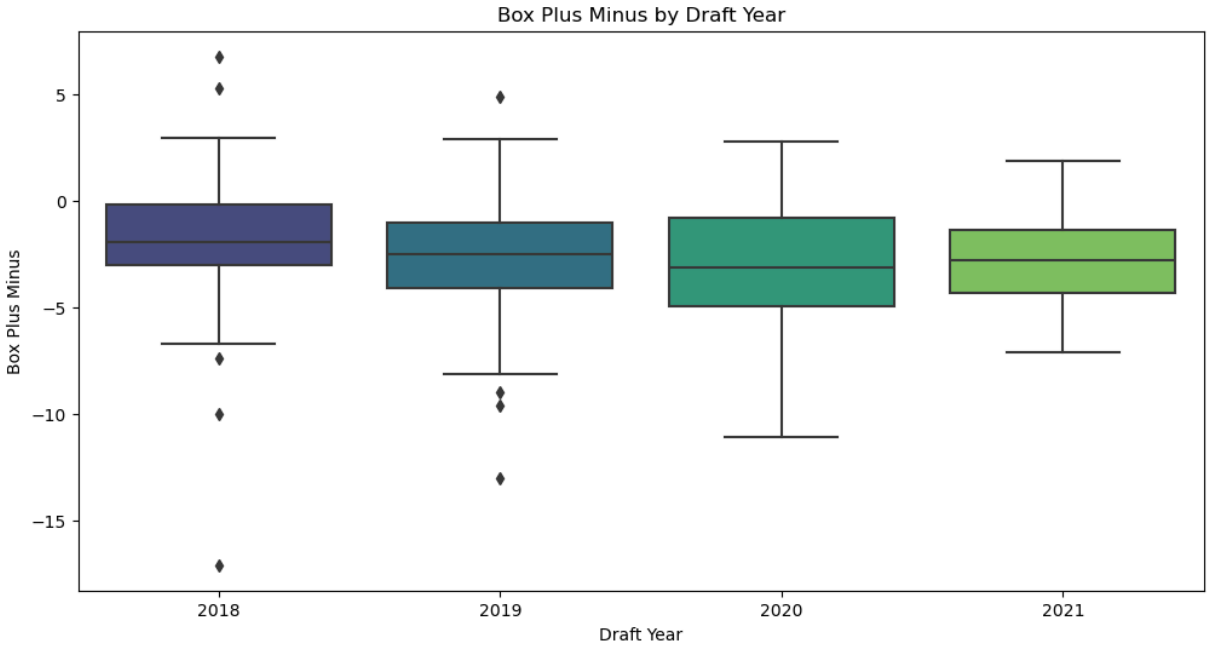



Career Longevity by Draft Year
Years Active
4.0
3.5
3.0
2.5
2.0
1.5
1.0
2018
2019
2020
2021
Draft Year


Average Total Rebounds by Draft Year
Average Total Rebounds
10
8
6
4
2
0
2018
2019
2020
2021
Draft Year


Average Assists by Draft Year
Average Assists
0
2
4
6
8
2018
2019
2020
2021
Draft Year


Win Shares by Draft Year
Win Shares
0
5
10
15
20
25
30
2018
2019
2020
2021
Draft Year



Filter for Players from UNC

In [34]:
```
# Filter for players from UNC
unc_players = nba_draft_data[nba_draft_data['college'] == 'UNC']

# Display the result
print(unc_players)
```

	id	year	rank	overall_pick	team	player	college	\
4	5	1989	5	5	CHH	J.R. Reid	UNC	
131	132	1991	24	24	BOS	Rick Fox	UNC	
134	135	1991	27	27	SAC	Pete Chilcutt	UNC	
181	182	1992	20	20	NYK	Hubert Davis	UNC	
227	228	1993	12	12	LAL	George Lynch	UNC	
278	279	1994	9	9	BOS	Eric Montross	UNC	
326	327	1995	3	3	PHI	Jerry Stackhouse	UNC	
327	328	1995	4	4	WSB	Rasheed Wallace	UNC	
418	419	1996	37	37	DEN	Jeff McInnis	UNC	
500	501	1998	4	4	TOR	Antawn Jamison	UNC	
501	502	1998	5	5	GSW	Vince Carter	UNC	
530	531	1998	34	34	CHI	Shammond Williams	UNC	
690	691	2001	20	20	CLE	Brendan Haywood	UNC	
691	692	2001	21	21	BOS	Joseph Forte	UNC	
903	904	2005	2	2	ATL	Marvin Williams	UNC	
906	907	2005	5	5	CHA	Raymond Felton	UNC	
914	915	2005	13	13	CHA	Sean May	UNC	
915	916	2005	14	14	MIN	Rashad McCants	UNC	
1000	1001	2006	39	39	MIL	David Noel	UNC	
1029	1030	2007	8	8	CHA	Brandan Wright	UNC	
1154	1155	2009	13	13	IND	Tyler Hansbrough	UNC	
1159	1160	2009	18	18	MIN	Ty Lawson	UNC	
1169	1170	2009	28	28	MIN	Wayne Ellington	UNC	
1187	1188	2009	46	46	CLE	Danny Green	UNC	
1214	1215	2010	13	13	TOR	Ed Davis	UNC	
1328	1329	2012	7	7	GSW	Harrison Barnes	UNC	
1334	1335	2012	13	13	PHO	Kendall Marshall	UNC	
1335	1336	2012	14	14	MIL	John Henson	UNC	
1338	1339	2012	17	17	DAL	Tyler Zeller	UNC	
1406	1407	2013	25	25	LAC	Reggie Bullock	UNC	
1467	1468	2014	26	26	MIA	P.J. Hairston	UNC	
1586	1587	2016	25	25	LAC	Brice Johnson	UNC	
1616	1617	2016	55	55	BRK	Marcus Paige	UNC	
1636	1637	2017	15	15	POR	Justin Jackson	UNC	
1649	1650	2017	28	28	LAL	Tony Bradley	UNC	
1748	1749	2019	7	7	CHI	Coby White	UNC	
1752	1753	2019	11	11	MIN	Cameron Johnson	UNC	
1766	1767	2019	25	25	POR	Nassir Little	UNC	
1816	1817	2020	15	15	ORL	Cole Anthony	UNC	
1890	1891	2021	29	29	PHO	Day'Ron Sharpe	UNC	

	years_active	games	minutes_played	...	free_throw_percentage	\
4	11.0	672.0	15370.0	...	0.716	
131	13.0	930.0	23723.0	...	0.770	
134	9.0	584.0	8394.0	...	0.696	
181	12.0	685.0	15143.0	...	0.837	
227	12.0	774.0	17683.0	...	0.652	
278	8.0	465.0	8479.0	...	0.478	
326	18.0	970.0	30222.0	...	0.822	
327	16.0	1109.0	36243.0	...	0.721	
418	11.0	576.0	15927.0	...	0.796	
500	16.0	1083.0	37638.0	...	0.724	
501	22.0	1541.0	46367.0	...	0.798	
530	7.0	325.0	5351.0	...	0.798	
690	13.0	816.0	18680.0	...	0.587	

691	2.0	25.0	125.0	...	0.800
903	15.0	1072.0	30159.0	...	0.808
906	14.0	971.0	28829.0	...	0.790
914	4.0	119.0	1868.0	...	0.746
915	4.0	249.0	5037.0	...	0.741
1000	1.0	68.0	792.0	...	0.860
1029	10.0	428.0	6949.0	...	0.674
1154	7.0	428.0	7233.0	...	0.738
1159	8.0	551.0	16088.0	...	0.770
1169	13.0	770.0	16100.0	...	0.843
1187	13.0	819.0	20743.0	...	0.804
1214	12.0	722.0	13756.0	...	0.583
1328	10.0	747.0	23763.0	...	0.804
1334	4.0	160.0	3083.0	...	0.611
1335	8.0	445.0	8752.0	...	0.568
1338	8.0	414.0	7254.0	...	0.764
1406	9.0	434.0	9775.0	...	0.842
1467	2.0	111.0	2000.0	...	0.810
1586	2.0	21.0	107.0	...	0.600
1616	1.0	5.0	28.0	...	1.000
1636	5.0	255.0	4779.0	...	0.799
1649	5.0	167.0	1961.0	...	0.672
1748	3.0	195.0	5505.0	...	0.851
1752	3.0	183.0	4422.0	...	0.843
1766	3.0	138.0	2301.0	...	0.727
1816	2.0	112.0	3332.0	...	0.846
1890	1.0	32.0	391.0	...	0.585

	average_minutes_played	points_per_game	average_total_rebounds	\
4	22.9	8.5	5.0	
131	25.5	9.6	3.8	
134	14.4	4.3	3.3	
181	22.1	8.2	1.5	
227	22.8	6.6	5.0	
278	18.2	4.5	4.6	
326	31.2	16.9	3.2	
327	32.7	14.4	6.7	
418	27.7	9.4	2.0	
500	34.8	18.5	7.5	
501	30.1	16.7	4.3	
530	16.5	5.8	1.6	
690	22.9	6.8	6.0	
691	5.0	1.2	0.7	
903	28.1	10.2	5.2	
906	29.7	11.2	3.0	
914	15.7	6.9	4.0	
915	20.2	10.0	2.0	
1000	11.6	2.7	1.8	
1029	16.2	7.0	3.6	
1154	16.9	6.7	4.2	
1159	29.2	12.7	2.7	
1169	20.9	8.0	2.1	
1187	25.3	8.7	3.4	
1214	19.1	5.9	6.4	
1328	31.8	14.1	5.1	
1334	19.3	5.0	1.6	

1335	19.7	7.6	5.3
1338	17.5	6.9	4.4
1406	22.5	7.8	2.5
1467	18.0	6.0	2.4
1586	5.1	2.2	1.6
1616	5.6	2.4	0.8
1636	18.7	6.5	2.5
1649	11.7	4.6	4.3
1748	28.2	13.7	3.6
1752	24.2	10.4	3.6
1766	16.7	5.8	3.4
1816	29.8	14.9	5.1
1890	12.2	6.2	5.0

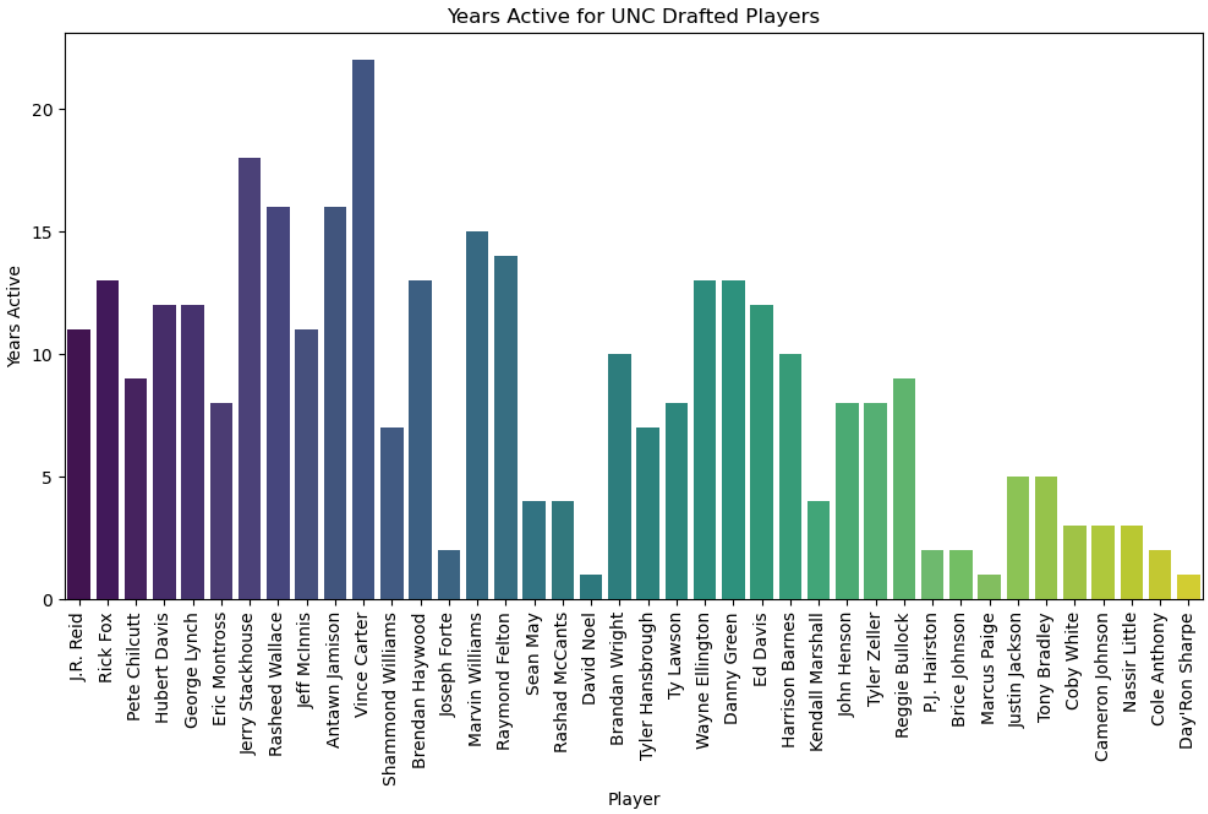
	average_assists	win_shares	win_shares_per_48_minutes	box_plus_minus
4	1.0	22.5	0.070	-2.9
131	2.8	44.7	0.090	0.0
134	0.8	10.6	0.060	-1.8
181	1.7	27.7	0.088	-0.9
227	1.4	32.3	0.088	-0.4
278	0.4	8.5	0.048	-4.4
326	3.3	52.4	0.083	0.3
327	1.8	105.1	0.139	2.2
418	4.4	23.1	0.070	-2.1
500	1.6	87.8	0.112	0.6
501	3.1	125.3	0.130	3.0
530	2.4	8.8	0.079	-0.7
690	0.5	43.8	0.113	-1.4
691	0.7	-0.4	-0.166	-8.6
903	1.3	65.1	0.104	-0.4
906	5.2	40.2	0.067	-0.6
914	1.0	2.6	0.068	-1.7
915	1.3	3.6	0.035	-1.7
1000	1.0	0.4	0.024	-3.1
1029	0.5	26.2	0.181	2.3
1154	0.4	20.2	0.134	-2.0
1159	6.0	42.1	0.126	1.4
1169	1.1	21.1	0.063	-1.5
1187	1.6	49.1	0.114	1.9
1214	0.7	43.3	0.151	-0.2
1328	1.8	44.8	0.091	-0.8
1334	4.9	1.0	0.015	-3.5
1335	1.1	20.7	0.113	-0.5
1338	0.9	17.1	0.113	-2.0
1406	1.2	18.1	0.089	-0.8
1467	0.5	0.9	0.021	-3.9
1586	0.1	0.1	0.062	-0.6
1616	0.6	0.0	-0.006	-8.8
1636	1.1	5.3	0.053	-2.8
1649	0.5	7.4	0.181	0.2
1748	3.5	5.6	0.049	-2.5
1752	1.4	10.9	0.118	1.1
1766	0.7	3.4	0.071	-2.3
1816	5.0	1.6	0.023	-2.0
1890	0.5	1.2	0.152	-2.8

```
      value_over_replacement  Cluster
4                       -3.7        2
131                     11.7        0
134                      0.3        1
181                      4.2        1
227                      7.3        2
278                     -5.2        2
326                     17.4        0
327                     38.3        0
418                     -0.3        0
500                     24.5        0
501                     57.9        0
530                      1.7        1
690                      2.9        2
691                     -0.2        1
903                     12.3        0
906                     10.1        0
914                      0.2        1
915                      0.4        1
1000                    -0.2        1
1029                     7.5        2
1154                     0.0        2
1159                    13.6        0
1169                     2.2        1
1187                    20.2        0
1214                     6.2        2
1328                     7.1        0
1334                    -1.1        1
1335                     3.4        2
1338                     0.0        2
1406                     3.0        1
1467                    -1.0        1
1586                     0.0        2
1616                     0.0        1
1636                    -1.0        1
1649                     1.1        2
1748                    -0.6        0
1752                     3.5        1
1766                    -0.2        1
1816                     0.0        0
1890                    -0.1        2

[40 rows x 25 columns]
```

Years Active for UNC Drafted Players

In [35]:
```
# Create a bar chart
plt.figure(figsize=(12, 6))
sns.barplot(data=unc_players, x='player', y='years_active', palette='viridis

# Set plot title and labels
plt.title('Years Active for UNC Drafted Players')
plt.xlabel('Player')
```

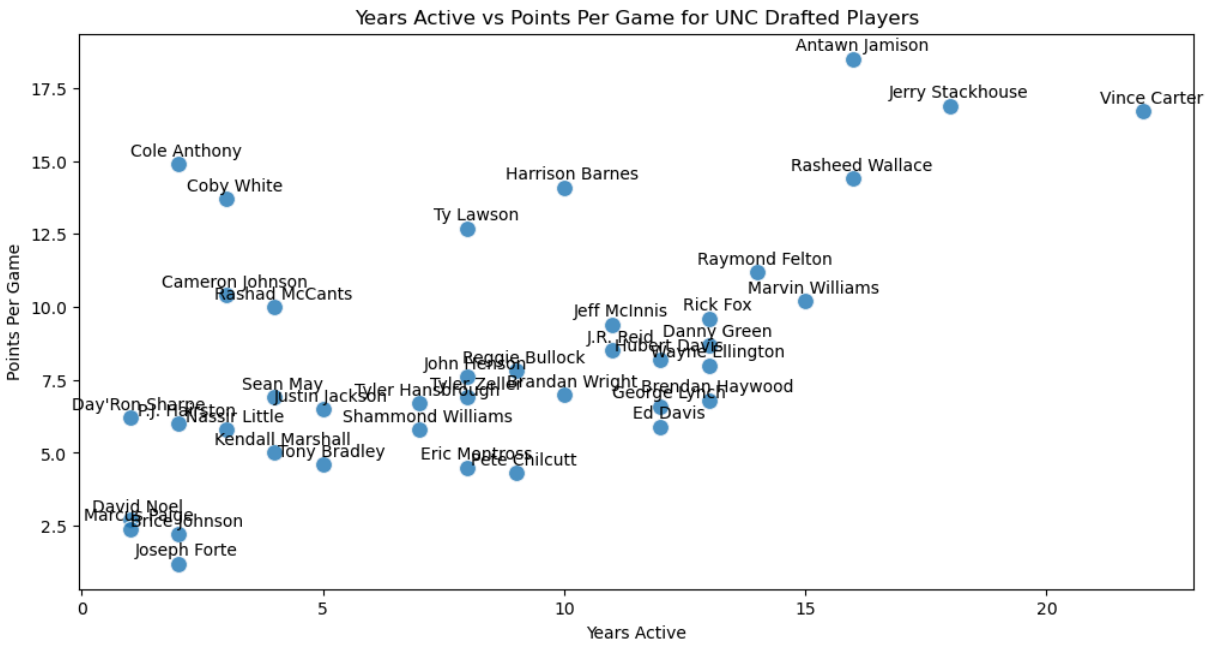
```
plt.ylabel('Years Active')
plt.xticks(rotation=90)  # Rotate the x labels for better readability

# Show the plot
plt.show()
```



Years Active vs Points Per Game for UNC Drafted Players

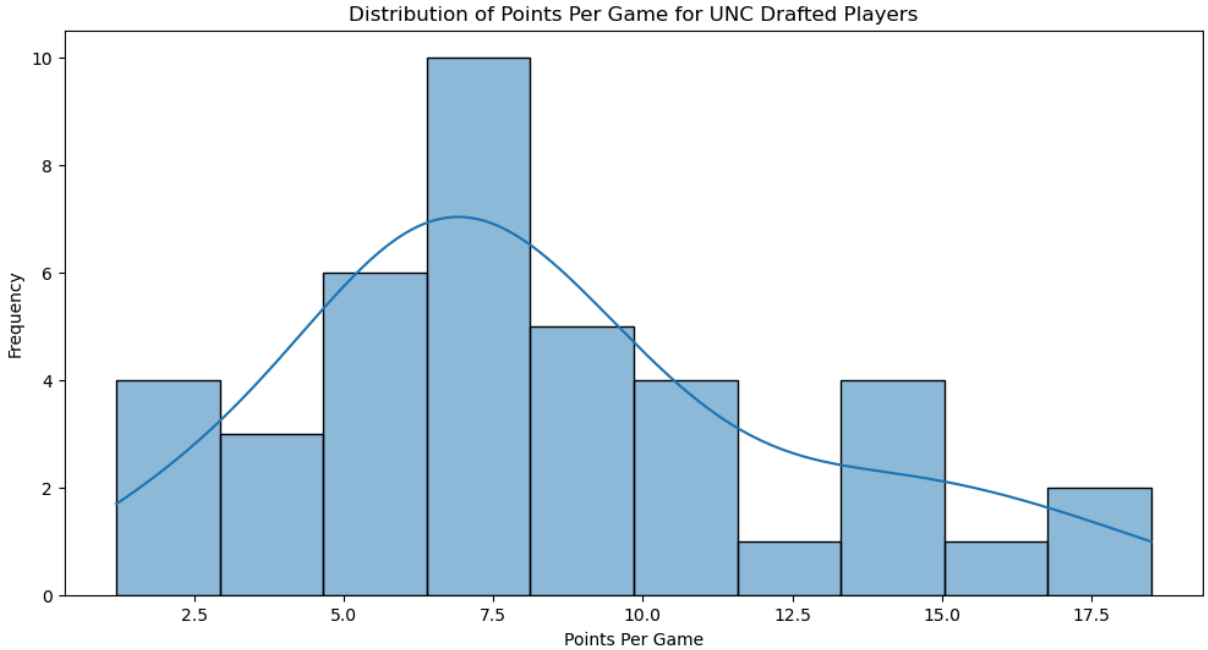
In [36]:
```
# Filter for players from UNC
unc_players = nba_draft_data[nba_draft_data['college'] == 'UNC']

# Create a scatter plot
plt.figure(figsize=(12, 6))
scatter = sns.scatterplot(data=unc_players, x='years_active', y='points_per_

# Annotate each point with the player's name
for i in range(unc_players.shape[0]):
    plt.annotate(unc_players['player'].iloc[i],
                 (unc_players['years_active'].iloc[i], unc_players['points_p
                 textcoords="offset points", xytext=(5,5), ha='center')

# Set plot title and labels
plt.title('Years Active vs Points Per Game for UNC Drafted Players')
plt.xlabel('Years Active')
plt.ylabel('Points Per Game')
```

```
# Show the plot
plt.show()
```



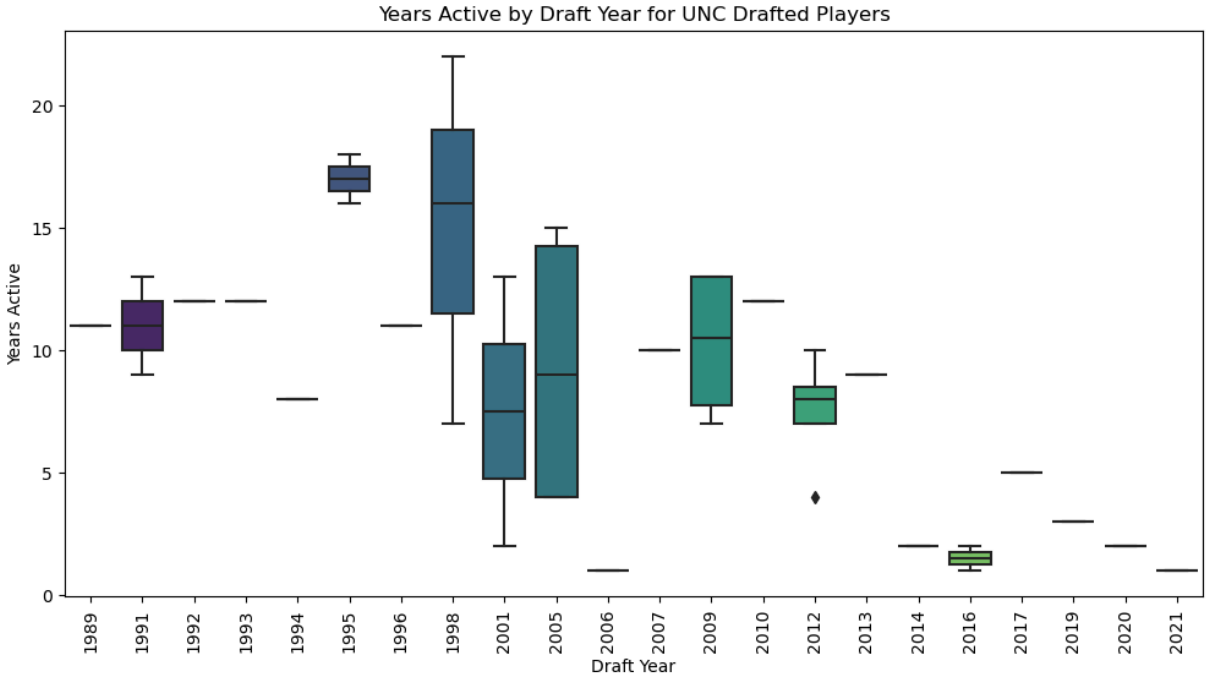
In [37]:
```
plt.figure(figsize=(12, 6))
sns.histplot(unc_players['points_per_game'], bins=10, kde=True)
plt.title('Distribution of Points Per Game for UNC Drafted Players')
plt.xlabel('Points Per Game')
plt.ylabel('Frequency')
plt.show()
```

```
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



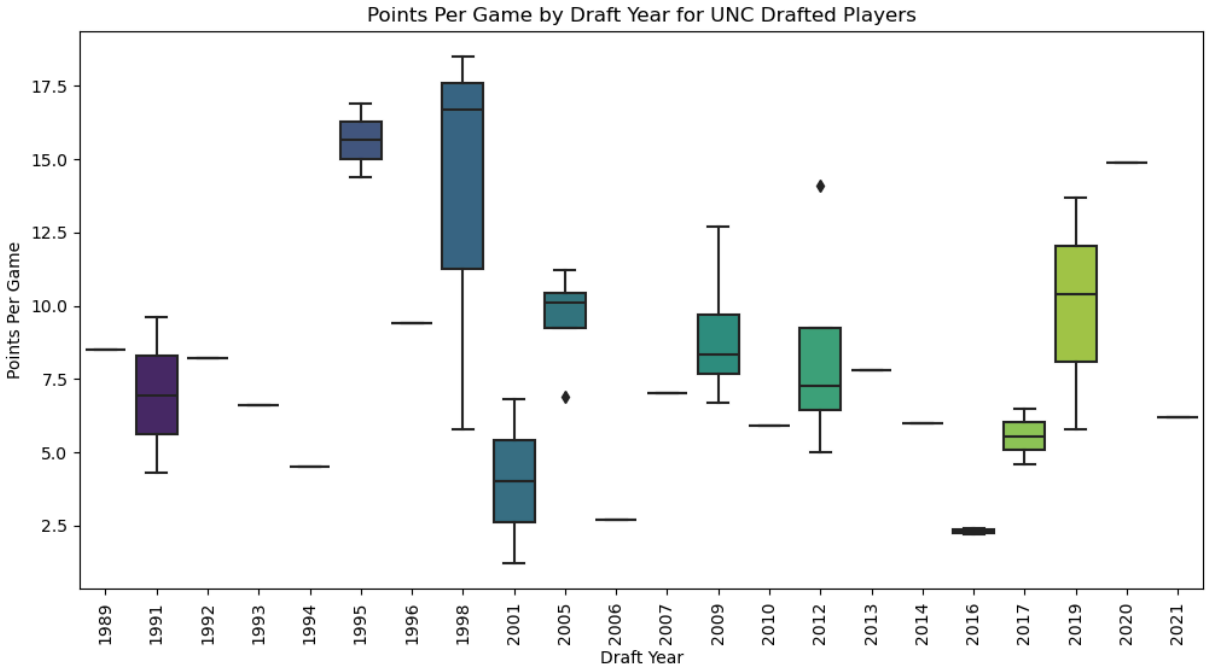
Years Active by Draft Year for UNC Drafted Players

In [38]:
```
plt.figure(figsize=(12, 6))
sns.boxplot(data=unc_players, x='year', y='years_active', palette='viridis')
plt.title('Years Active by Draft Year for UNC Drafted Players')
plt.xlabel('Draft Year')
plt.ylabel('Years Active')
plt.xticks(rotation=90)  # Rotate x labels for better readability
plt.show()
```



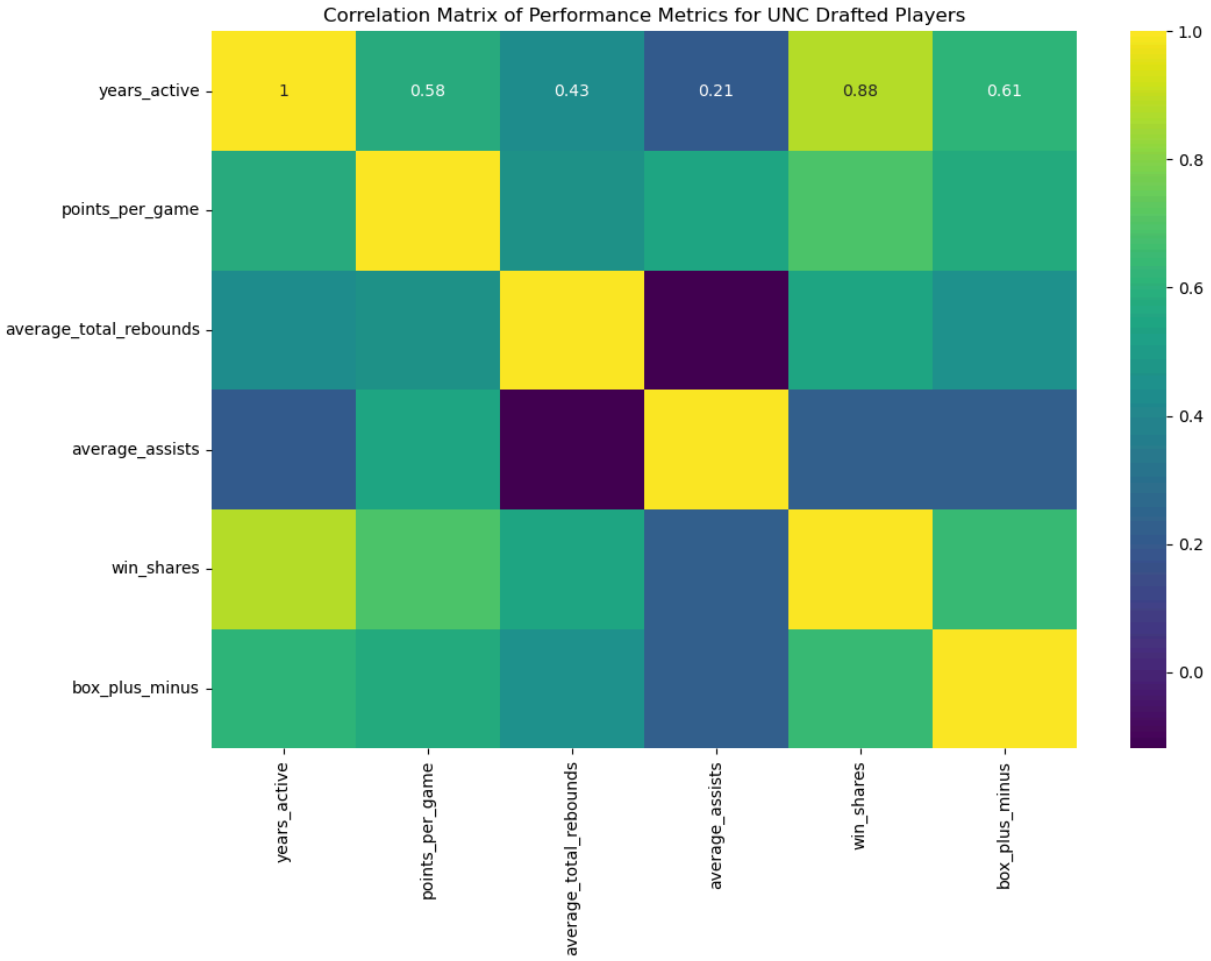
Points Per Game by Draft Year for UNC Drafted Players

In [39]:
```
plt.figure(figsize=(12, 6))
sns.boxplot(data=unc_players, x='year', y='points_per_game', palette='viridi
plt.title('Points Per Game by Draft Year for UNC Drafted Players')
plt.xlabel('Draft Year')
plt.ylabel('Points Per Game')
plt.xticks(rotation=90)  # Rotate x labels for better readability
plt.show()
```



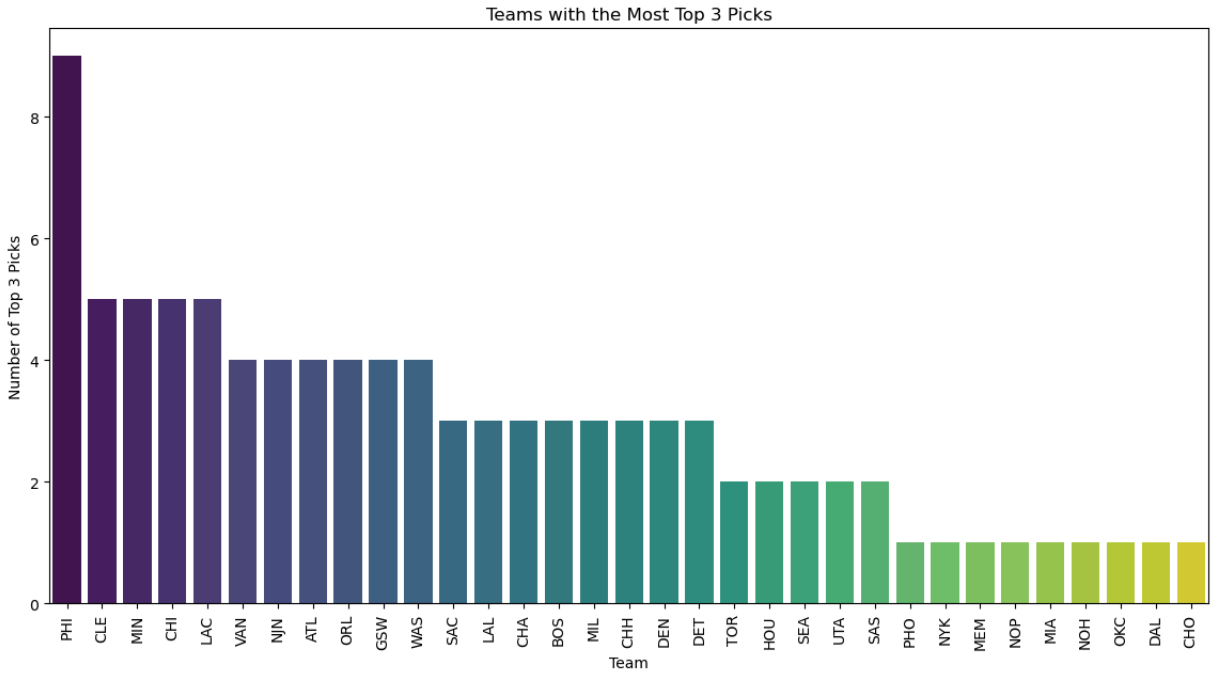
Correlation Matrix of Performance Metrics for UNC Drafted Players

In [40]:
```
plt.figure(figsize=(12, 8))
correlation_matrix = unc_players[['years_active', 'points_per_game', 'averag
sns.heatmap(correlation_matrix, annot=True, cmap='viridis')
plt.title('Correlation Matrix of Performance Metrics for UNC Drafted Players
plt.show()
```



Teams with the Most Top 3 Picks since 1989

In [41]:
```
# Filter for top 3 picks
top_3_picks = nba_draft_data[nba_draft_data['rank'] <= 3]

# Count the occurrences for each team
top_3_picks_counts = top_3_picks['team'].value_counts()

# Convert to a DataFrame for easier plotting
top_3_picks_counts_df = top_3_picks_counts.reset_index()
top_3_picks_counts_df.columns = ['team', 'top_3_picks_count']

# Plot the results
plt.figure(figsize=(14, 7))
sns.barplot(data=top_3_picks_counts_df, x='team', y='top_3_picks_count', pal
plt.title('Teams with the Most Top 3 Picks')
plt.xlabel('Team')
plt.ylabel('Number of Top 3 Picks')
plt.xticks(rotation=90)
plt.show()
```



In [42]:
```
top_3_picks = nba_draft_data[nba_draft_data['overall_pick'] <= 3]
```

Average performance metrics for top 3 picks

In [43]:
```
# Calculate the average performance metrics for top 3 picks
top_3_metrics = top_3_picks[['points_per_game', 'years_active', 'win_shares'
print("Average Performance Metrics for Top 3 Picks:")
print(top_3_metrics)
```

```
Average Performance Metrics for Top 3 Picks:
points_per_game    15.500000
years_active       10.479167
win_shares         55.115625
box_plus_minus      1.026042
dtype: float64
```

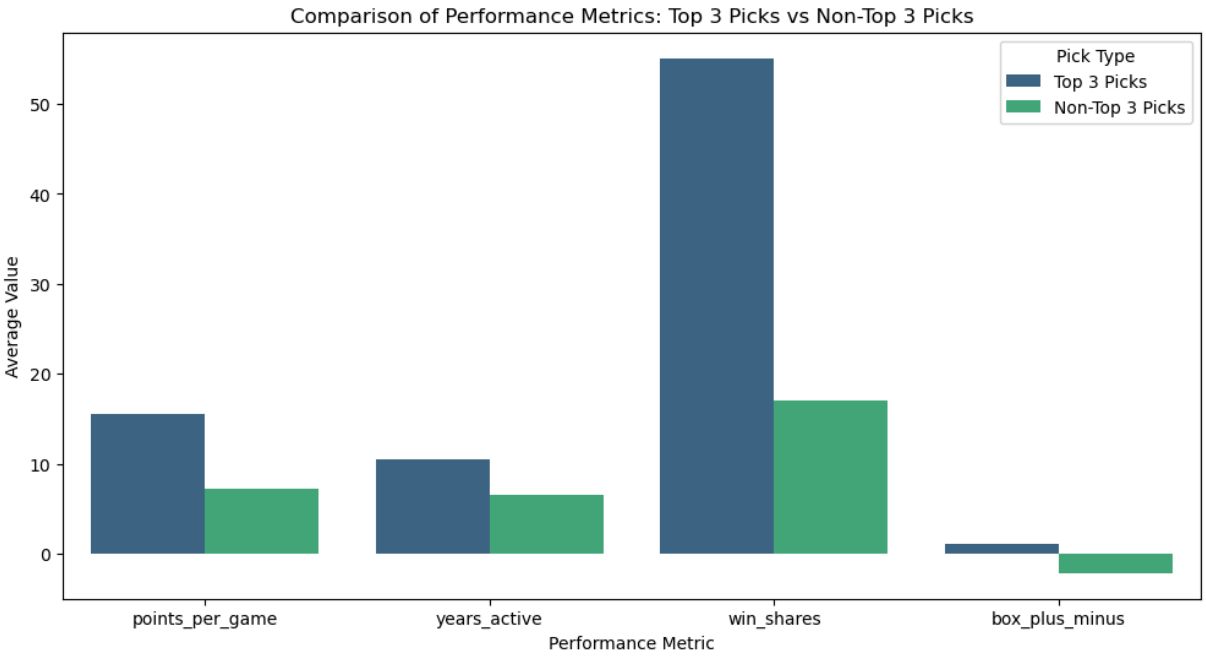
Average performance metrics for non-top 3 picks

In [44]:
```
# Calculate the average performance metrics for non-top 3 picks
non_top_3_picks = nba_draft_data[nba_draft_data['overall_pick'] > 3]
non_top_3_metrics = non_top_3_picks[['points_per_game', 'years_active', 'wir
print("Average Performance Metrics for Non-Top 3 Picks:")
print(non_top_3_metrics)
```

```
Average Performance Metrics for Non-Top 3 Picks:
points_per_game     7.210816
years_active        6.477320
win_shares         17.003280
box_plus_minus     -2.140475
dtype: float64
```

Comparison of Performance Metrics: Top 3 Picks vs Non-Top 3 Picks

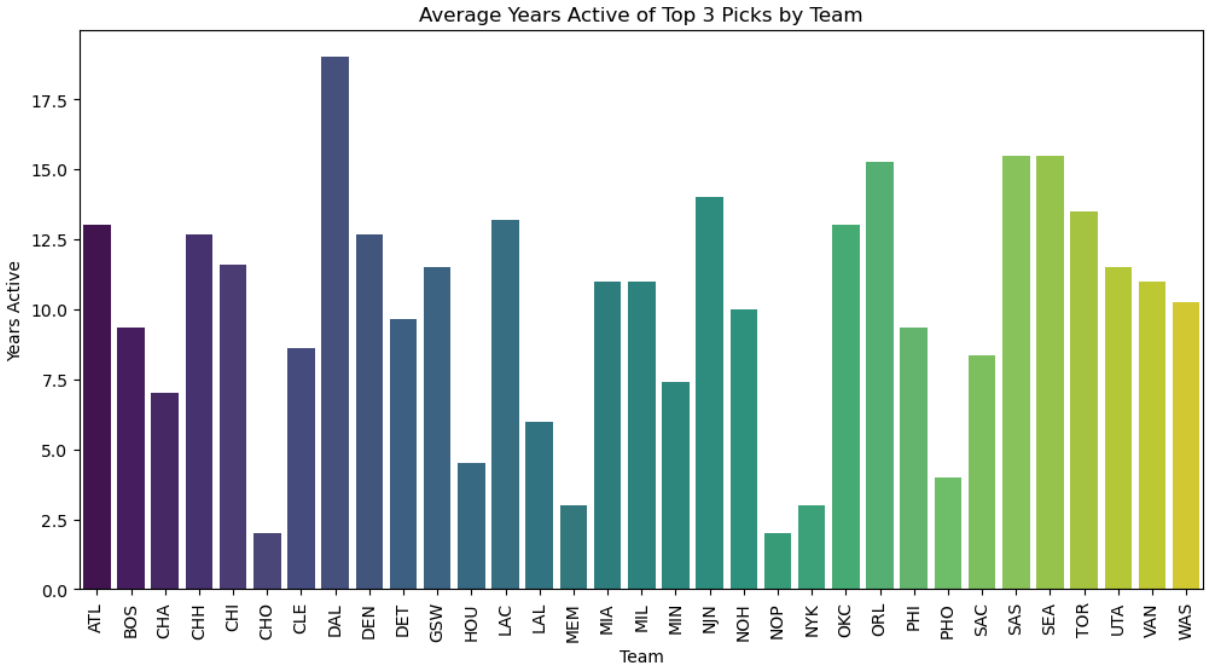
In [45]:
```
# Combine the metrics into a single DataFrame for plotting
metrics_df = pd.DataFrame({
    'Top 3 Picks': top_3_metrics,
    'Non-Top 3 Picks': non_top_3_metrics
}).reset_index().melt(id_vars='index', var_name='Pick Type', value_name='Met

# Plot the comparison of performance metrics
plt.figure(figsize=(12, 6))
sns.barplot(data=metrics_df, x='index', y='Metric Value', hue='Pick Type', p
plt.title('Comparison of Performance Metrics: Top 3 Picks vs Non-Top 3 Picks
plt.xlabel('Performance Metric')
plt.ylabel('Average Value')
plt.legend(title='Pick Type')
plt.show()
```

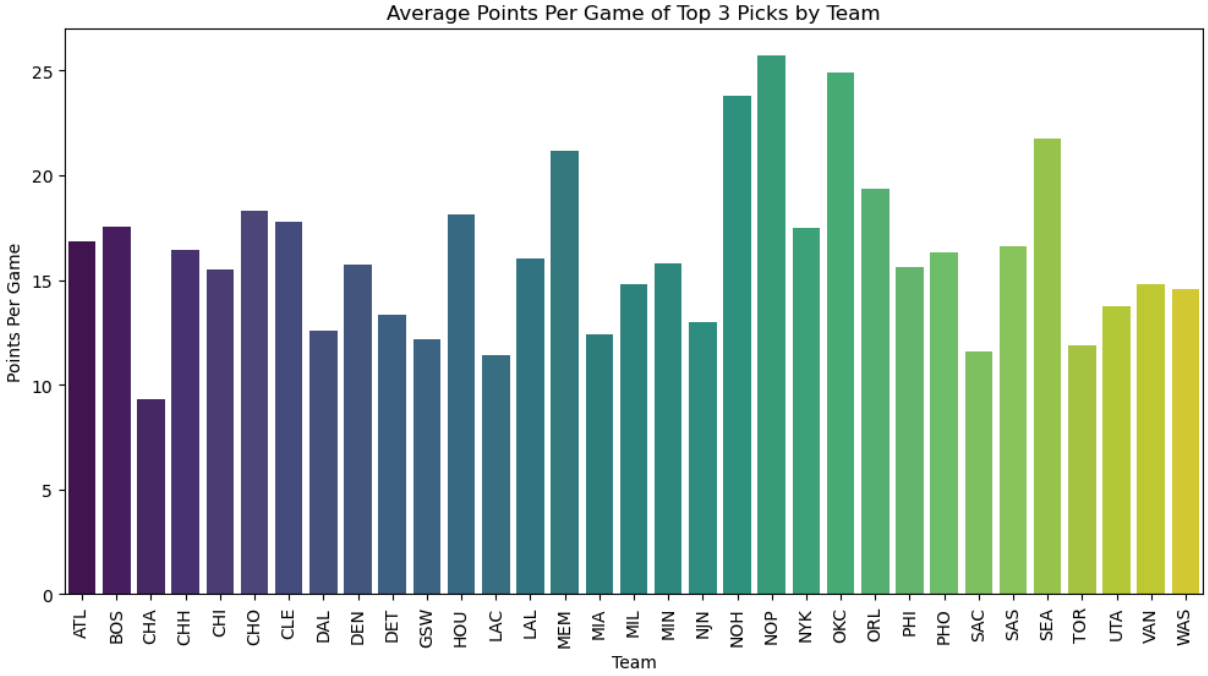


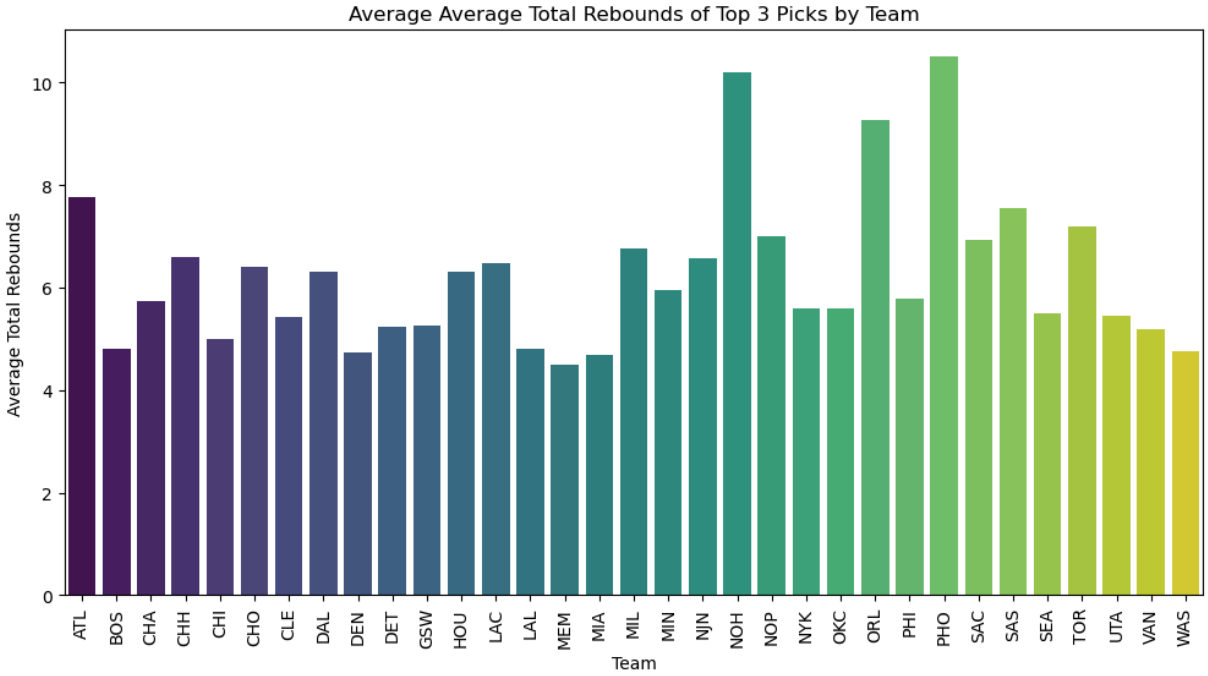
Performance metrics

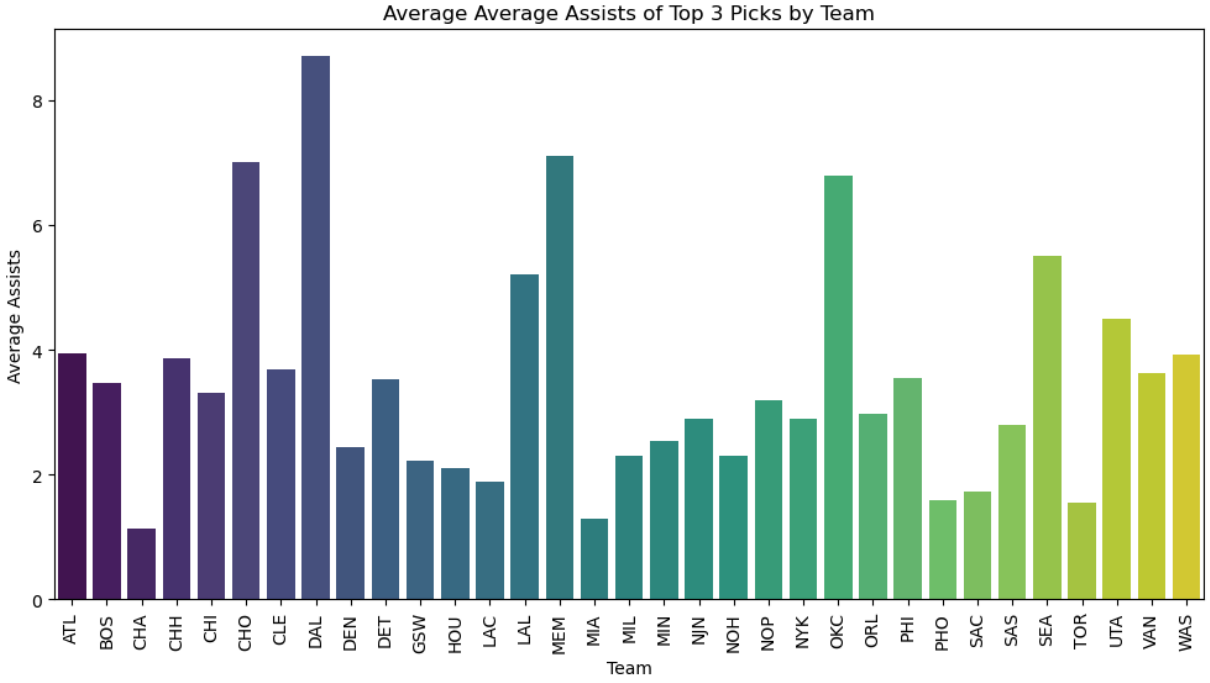
In [46]:
```
# Filter for top 3 picks
top_3_picks = nba_draft_data[nba_draft_data['overall_pick'] <= 3]

# Performance metrics to analyze
```

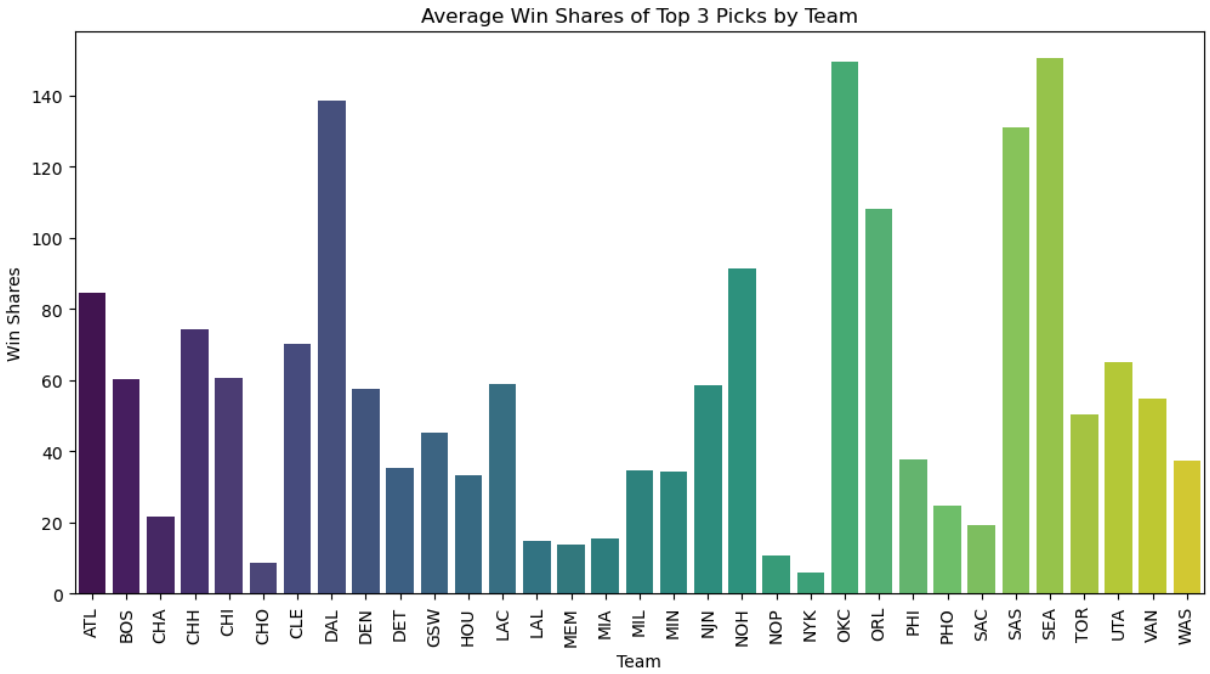
```
metrics = ['years_active', 'points_per_game', 'average_total_rebounds', 'ave

# Calculate average performance metrics for each team
team_performance = top_3_picks.groupby('team')[metrics].mean().reset_index()

# Visualize the performance metrics
def visualize_performance(metrics):
    for metric in metrics:
        plt.figure(figsize=(12, 6))
        sns.barplot(data=team_performance, x='team', y=metric, palette='viri
        plt.title(f'Average {metric.replace("_", " ").title()} of Top 3 Pick
        plt.xlabel('Team')
        plt.ylabel(metric.replace("_", " ").title())
        plt.xticks(rotation=90)
        plt.show()

visualize_performance(metrics)
```

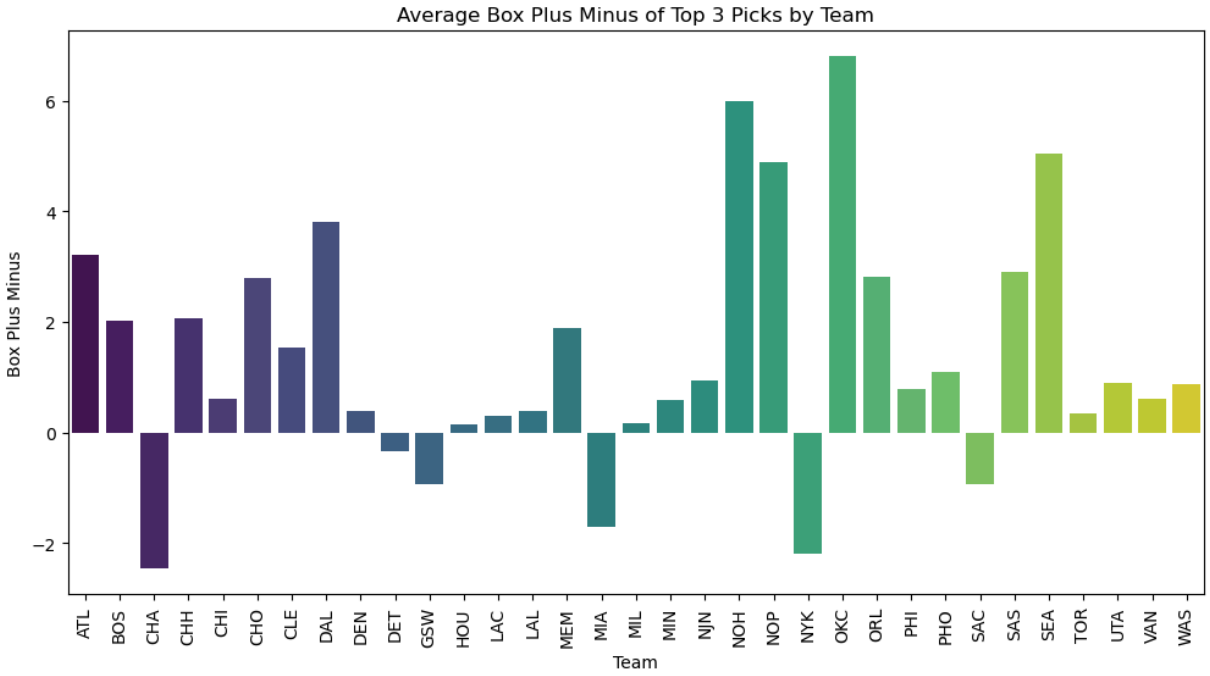



Average Points Per Game of Top 3 Picks by Team
Points Per Game
0
5
10
15
20
25
ATL BOS CHA CHH CHI CHO CLE DAL DEN DET GSW HOU LAC LAL MEM MIA MIL MIN NJN NOH NOP NYK OKC ORL PHI PHO SAC SAS SEA TOR UTA VAN WAS
Team


Average Average Total Rebounds of Top 3 Picks by Team
Average Total Rebounds
0
2
4
6
8
10
ATL BOS CHA CHH CHI CHO CLE DAL DEN DET GSW HOU LAC LAL MEM MIA MIL MIN NJN NOH NOP NYK OKC ORL PHI PHO SAC SAS SEA TOR UTA VAN WAS
Team

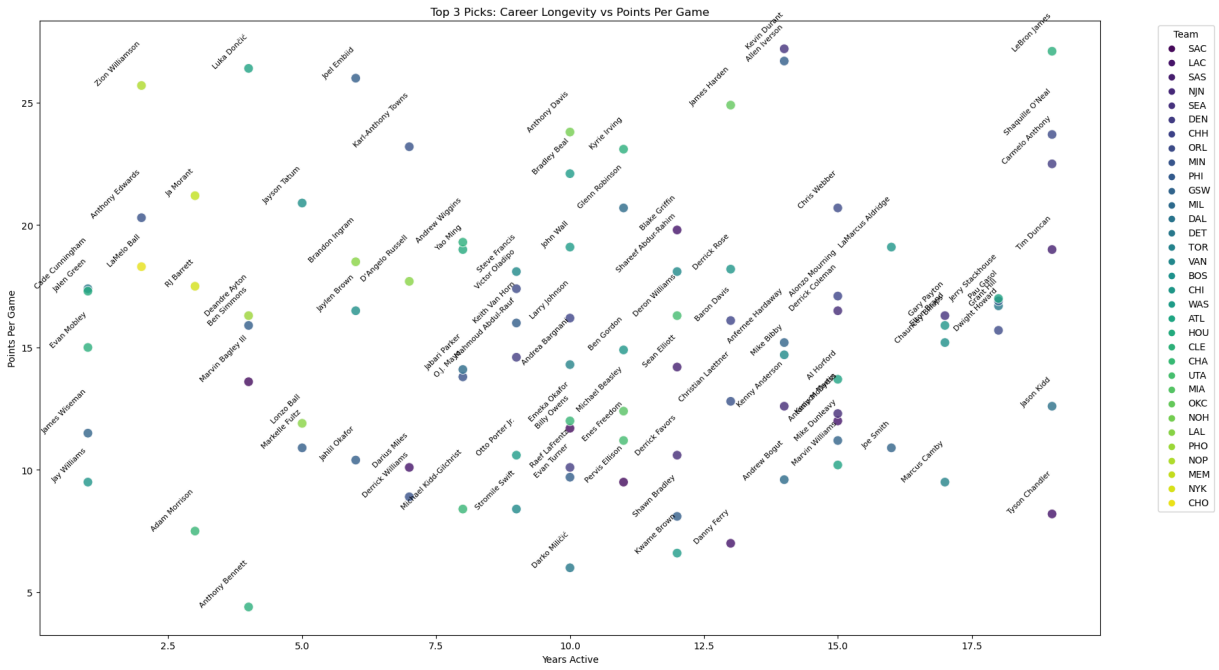

Average Average Assists of Top 3 Picks by Team
Average Assists
0
2
4
6
8
ATL BOS CHA CHH CHI CHO CLE DAL DEN DET GSW HOU LAC LAL MEM MIA MIL MIN NJN NOH NOP NYK OKC ORL PHI PHO SAC SAS SEA TOR UTA VAN WAS
Team


Average Win Shares of Top 3 Picks by Team
Win Shares
0
20
40
60
80
100
120
140
ATL BOS CHA CHH CHI CHO CLE DAL DEN DET GSW HOU LAC LAL MEM MIA MIL MIN NJN NOH NOP NYK OKC ORL PHI PHO SAC SAS SEA TOR UTA VAN WAS
Team



Top 3 Picks: Career Longevity vs Points Per Game

In [47]:
```
# Filter for top 3 picks
top_3_picks = nba_draft_data[nba_draft_data['overall_pick'] <= 3]

# Create a larger plot
plt.figure(figsize=(18, 10))
scatter = sns.scatterplot(
    data=top_3_picks,
    x='years_active',
    y='points_per_game',
    hue='team',
    palette='viridis',
    s=100,
    alpha=0.8
)

# Adjust text positions
for i, row in top_3_picks.iterrows():
    plt.text(row['years_active'], row['points_per_game'], row['player'], for

plt.title('Top 3 Picks: Career Longevity vs Points Per Game')
plt.xlabel('Years Active')
plt.ylabel('Points Per Game')
plt.legend(title='Team', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```

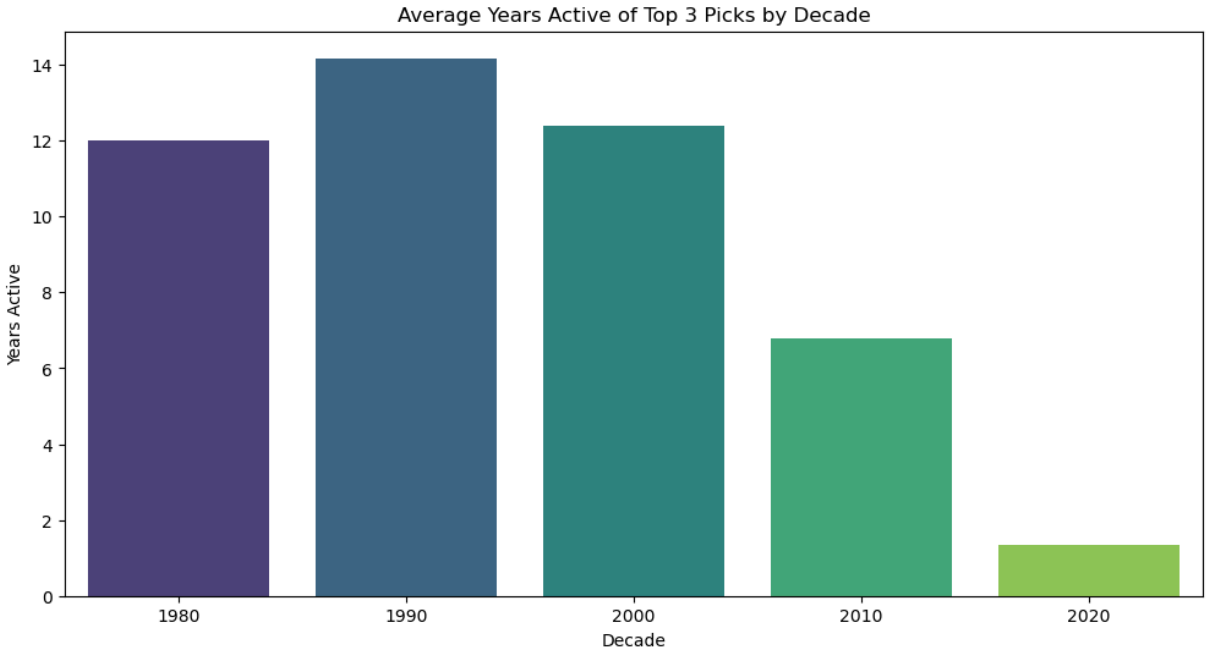


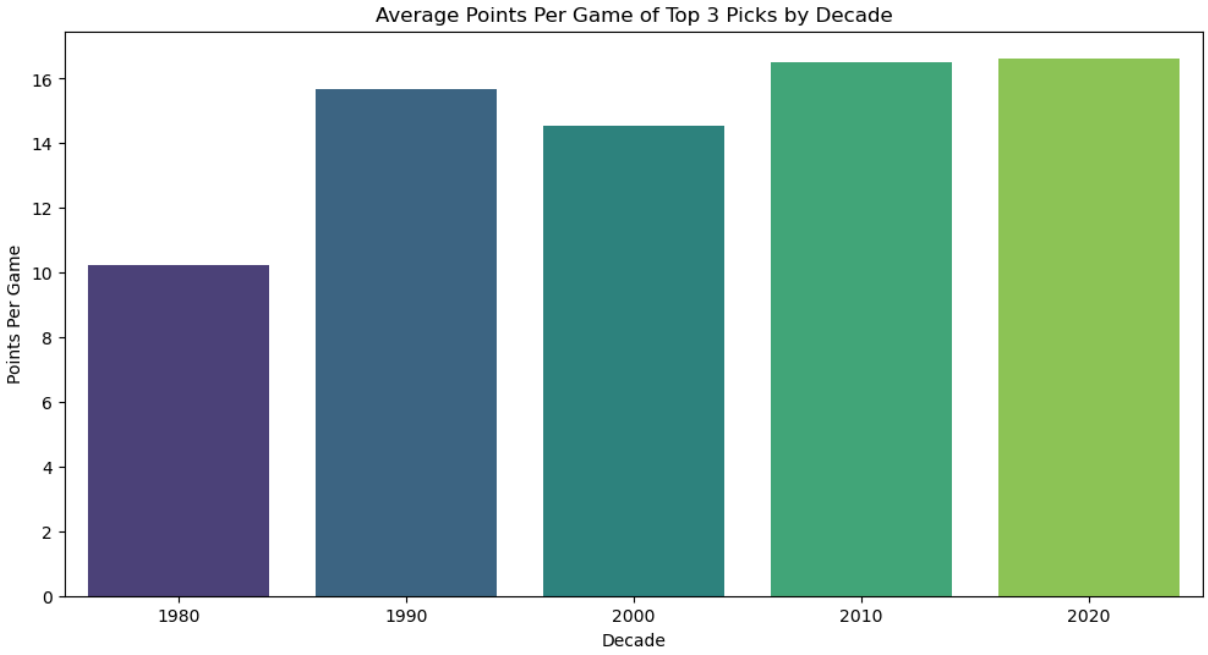
Performance metrics by decade

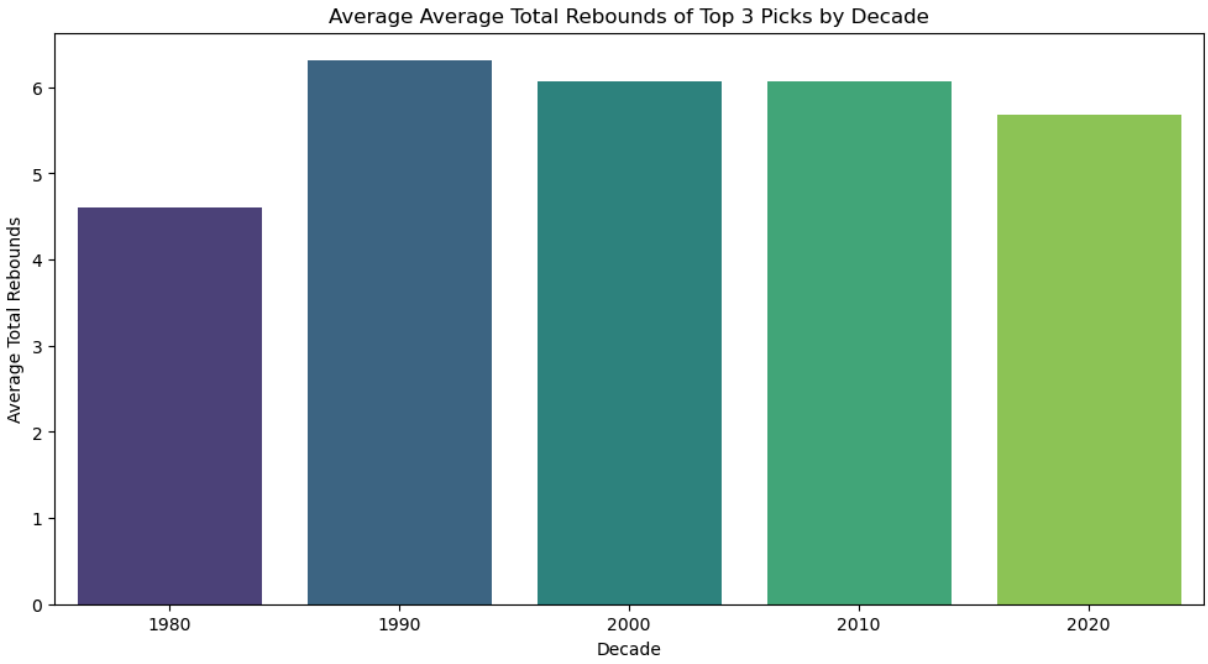
In [48]:
```
# Adding a decade column to the dataset
nba_draft_data['decade'] = (nba_draft_data['year'] // 10) * 10

# Filter for top 3 picks and add decade column
top_3_picks['decade'] = (top_3_picks['year'] // 10) * 10

# Calculate average performance metrics for each decade
decade_performance = top_3_picks.groupby('decade')[metrics].mean().reset_inc

# Visualize the performance metrics by decade
def visualize_decade_performance(metrics):
    for metric in metrics:
        plt.figure(figsize=(12, 6))
        sns.barplot(data=decade_performance, x='decade', y=metric, palette='
        plt.title(f'Average {metric.replace("_", " ").title()} of Top 3 Pick
        plt.xlabel('Decade')
        plt.ylabel(metric.replace("_", " ").title())
        plt.show()

visualize_decade_performance(metrics)

# Scatter plot for career longevity vs points per game by decade
plt.figure(figsize=(14, 8))
sns.scatterplot(
    data=top_3_picks,
    x='years_active',
    y='points_per_game',
    hue='decade',
    palette='viridis',
    s=100,
    alpha=0.8
```

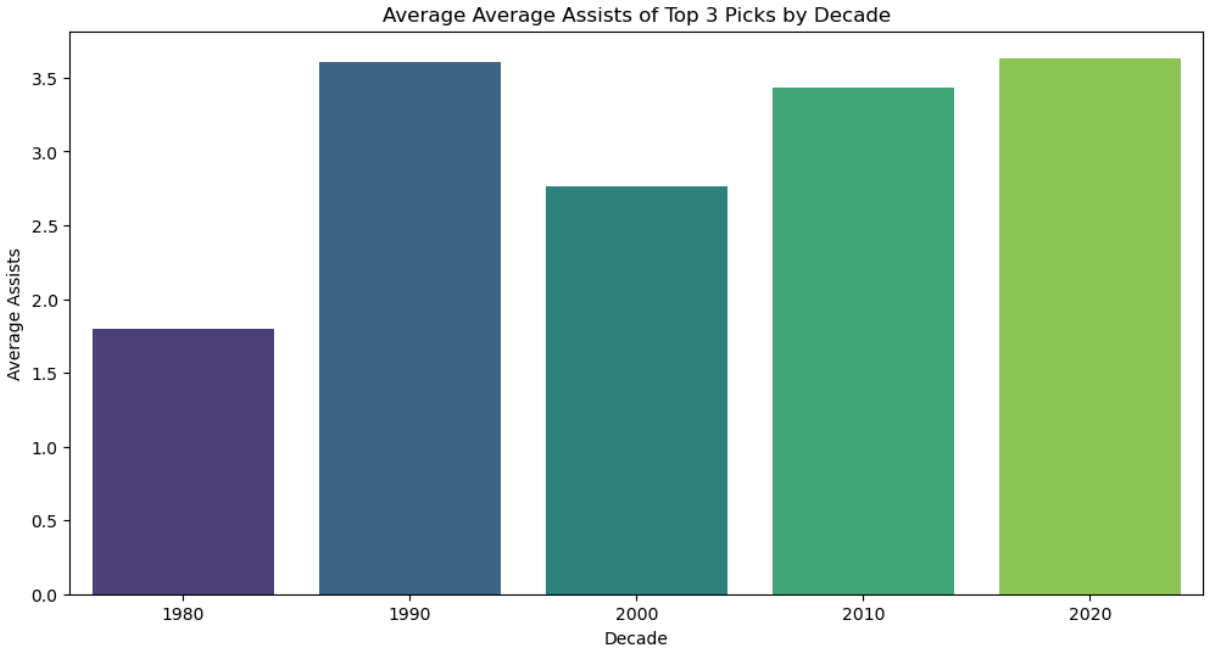
```
)

for i, row in top_3_picks.iterrows():
    plt.text(row['years_active'], row['points_per_game'], row['player'], for

plt.title('Top 3 Picks: Career Longevity vs Points Per Game by Decade')
plt.xlabel('Years Active')
plt.ylabel('Points Per Game')
plt.legend(title='Decade')
plt.tight_layout()
plt.show()
```

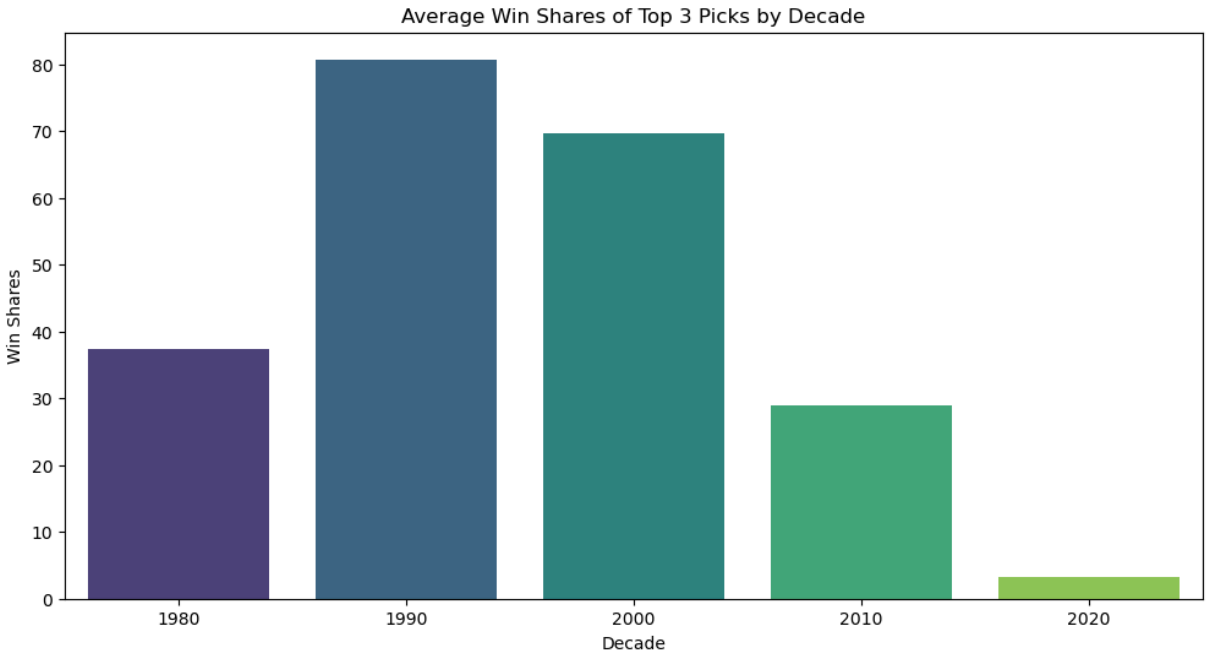
```
/var/folders/sh/8jrc1l412v3gtlzcz2bx_qcw0000gn/T/ipykernel_29956/1836839960.
py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  top_3_picks['decade'] = (top_3_picks['year'] // 10) * 10
```

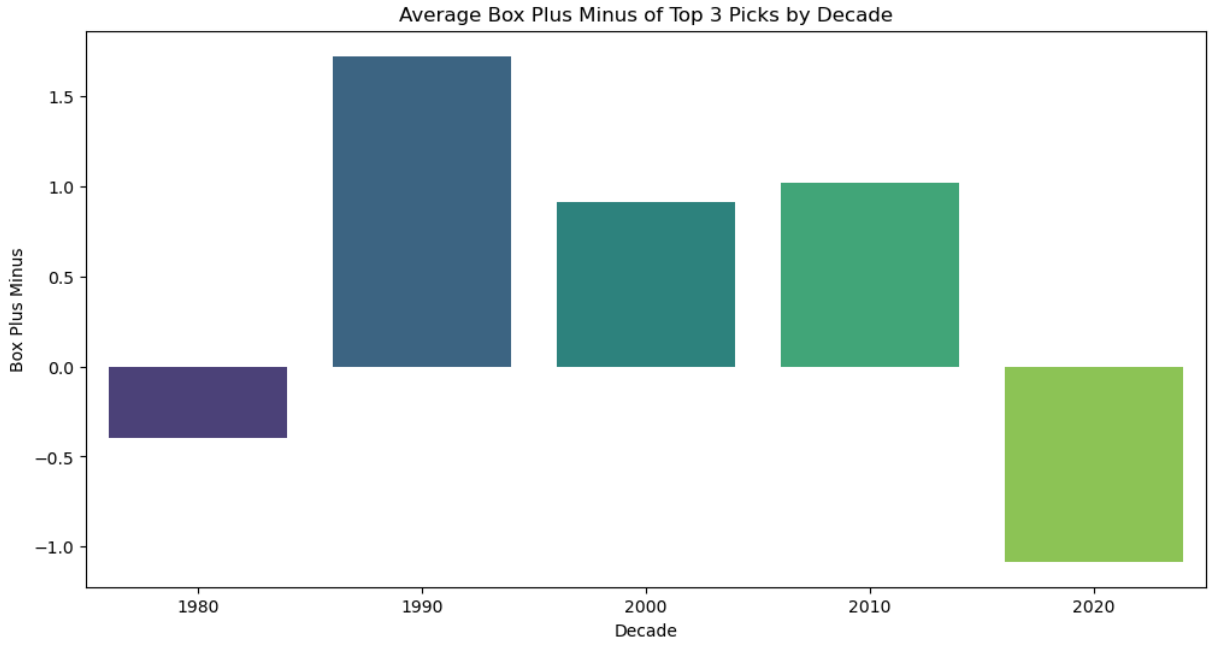


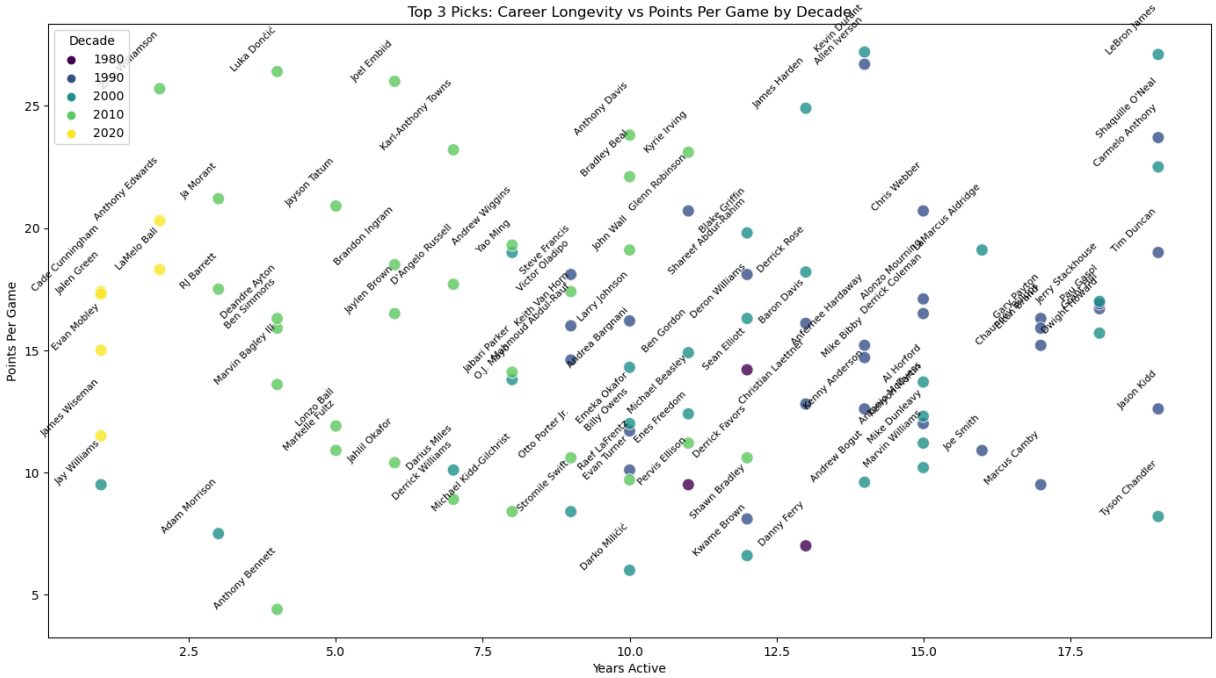

Average Points Per Game of Top 3 Picks by Decade
Points Per Game
0
2
4
6
8
10
12
14
16
1980
1990
2000
2010
2020
Decade


Average Average Total Rebounds of Top 3 Picks by Decade
Average Total Rebounds
0
1
2
3
4
5
6
1980
1990
2000
2010
2020
Decade


Average Average Assists of Top 3 Picks by Decade
Average Assists
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
1980
1990
2000
2010
2020
Decade


Average Win Shares of Top 3 Picks by Decade
Win Shares
0
10
20
30
40
50
60
70
80
1980
1990
2000
2010
2020
Decade





Win shares for top 5 players

In [49]:
```
# Filter for top 3 picks
top_3_picks = nba_draft_data[nba_draft_data['overall_pick'] <= 3]

# Sort by win shares and select the top 5 players
top_5_win_shares = top_3_picks.sort_values(by='win_shares', ascending=False)

# Display the top 5 players
print(top_5_win_shares[['player', 'year', 'team', 'win_shares', 'points_per_
```

	player	year	team	win_shares	points_per_game	years_active
785	LeBron James	2003	CLE	249.5	27.1	19.0
440	Tim Duncan	1997	SAS	206.4	19.0	19.0
162	Shaquille O'Neal	1992	ORL	181.7	23.7	19.0
1023	Kevin Durant	2007	SEA	155.2	27.2	14.0
1144	James Harden	2009	OKC	149.6	24.9	13.0