⛹️ Who are the Best Players in the NBA? ## **Project Overview:** This project analyses the **2024-2025 NBA Player Stats** to uncover insights into **individual performance trends** throughout the season. ## **Key Objectives:** 🏀 **Identify the best all-around players** in the NBA ⚖️ **Balance offensive and defensive metrics** to ensure fair evaluation 📊 **Explore key efficiency metrics and correlations** to gain deeper insights ## **Why is This Important?** 🔍 Helps **teams evaluate potential new talent objectively** 📈 Demonstrates how **data analysis impacts sports decision-making** 🗓️ *All data correct as of 05/03/2025* 📂 Click here to download NBA Player Stats

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from bokeh.plotting import figure, show, output_notebook
from bokeh.models import ColumnDataSource, HoverTool
from bokeh.transform import factor_cmap
from bokeh.palettes import Category10
from bokeh.io import push_notebook

import warnings
warnings.filterwarnings('ignore')

pd.set_option('display.max_columns', None)
```

Import & Clean Data

In [4]:
```
file_path = 'Data/NBA Stats.csv'

#Define column names while importing
column_names = [
    'name','team', 'current_team', 'position', 'age', 'games_played', 'minutes_per_game', 'usage_percent
    'turnover_percentage', 'free_throw_attempts', 'free_throw_percentage',
    'two_point_attempts', 'two_point_percentage', 'three_point_attempts',
    'three_point_percentage', 'effective_field_goal_percentage', 'true_shooting_percentage',
    'points_per_game', 'rebounds_per_game', 'assists_per_game', 'steals_per_game',
    'blocks_per_game', 'turnovers_per_game', 'points_plus_rebounds',
    'points_plus_assists', 'points_rebounds_assists', 'value_index',
    'offensive_rating', 'defensive_rating'
]

#Read CSV, rename columns and set the index to player name
df = pd.read_csv(file_path,
                 names=column_names,
                 header=0,
                index_col='name')

#Check for duplicates
duplicates_count = df.duplicated().sum()
print(f"Number of Duplicate Rows: {duplicates_count}")

#Check for null values
null_counts = df.isnull().sum()
print("Null Values per Column:")
print(null_counts)

display(df.head(2))
print('\nRows and Columns:', df.shape)
print(df.info())
display('Statistical Summary:', df.describe())
```

```
Number of Duplicate Rows: 0
Null Values per Column:
team                              0
current_team                      0
position                          0
age                               0
games_played                      0
minutes_per_game                  0
usage_percentage                  0
turnover_percentage               0
free_throw_attempts               0
free_throw_percentage             0
two_point_attempts                0
two_point_percentage              0
three_point_attempts              0
three_point_percentage            0
effective_field_goal_percentage   0
true_shooting_percentage          0
points_per_game                   0
rebounds_per_game                 0
assists_per_game                  0
steals_per_game                   0
blocks_per_game                   0
turnovers_per_game                0
points_plus_rebounds              0
points_plus_assists               0
points_rebounds_assists           0
value_index                       0
offensive_rating                  0
defensive_rating                  0
dtype: int64
```

name	team	current_team	position	age	games_played	minutes_per_game	usage_percentage	turnover_percent
Shai Gilgeous-Alexander	Okc	Yes	G	26.6	60	34.2	34.5	
Giannis Antetokounmpo	Mil	Yes	F	30.2	48	34.1	35.9	

```
Rows and Columns: (616, 28)
<class 'pandas.core.frame.DataFrame'>
Index: 616 entries, Shai Gilgeous-Alexander to Jahlil Okafor
Data columns (total 28 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   team                            616 non-null    object
 1   current_team                    616 non-null    object
 2   position                        616 non-null    object
 3   age                             616 non-null    float64
 4   games_played                    616 non-null    int64
 5   minutes_per_game                616 non-null    float64
 6   usage_percentage                616 non-null    float64
 7   turnover_percentage             616 non-null    float64
 8   free_throw_attempts             616 non-null    int64
 9   free_throw_percentage           616 non-null    float64
 10  two_point_attempts              616 non-null    int64
 11  two_point_percentage            616 non-null    float64
 12  three_point_attempts            616 non-null    int64
 13  three_point_percentage          616 non-null    float64
 14  effective_field_goal_percentage  616 non-null    float64
 15  true_shooting_percentage        616 non-null    float64
 16  points_per_game                 616 non-null    float64
 17  rebounds_per_game               616 non-null    float64
 18  assists_per_game                616 non-null    float64
 19  steals_per_game                 616 non-null    float64
 20  blocks_per_game                 616 non-null    float64
 21  turnovers_per_game              616 non-null    float64
 22  points_plus_rebounds            616 non-null    float64
 23  points_plus_assists             616 non-null    float64
 24  points_rebounds_assists         616 non-null    float64
 25  value_index                     616 non-null    float64
 26  offensive_rating                616 non-null    float64
 27  defensive_rating                616 non-null    float64
dtypes: float64(21), int64(4), object(3)
memory usage: 139.6+ KB
None
'Statistical Summary:'
```

	age	games_played	minutes_per_game	usage_percentage	turnover_percentage	free_throw_attempts	free_thr
count	616.000000	616.000000	616.000000	616.000000	616.000000	616.000000	
mean	26.584416	32.042208	18.701136	18.247565	13.896753	64.977273	
std	4.219072	19.653705	10.182105	6.530712	8.346379	82.088052	
min	19.200000	1.000000	0.900000	0.000000	0.000000	0.000000	
25%	23.500000	12.000000	9.500000	14.000000	9.975000	8.750000	
50%	25.700000	34.000000	18.650000	17.450000	13.000000	32.500000	
75%	29.000000	50.000000	26.900000	22.000000	16.750000	88.250000	
max	40.200000	63.000000	37.700000	49.300000	100.000000	537.000000	

Data Dictionary - NBA

Understanding the NBA Data Dictionary

This dataset contains detailed statistics for NBA players, covering various aspects of their performance, efficiency and impact on the game. Each column represents a key metric used by analysts, coaches and fans to evaluate players.

- **Basic Stats:** Information like team, position, age and games played.
- **Shooting Efficiency:** Metrics such as field goal percentages, effective field goal percentage and true shooting percentage.

- **Advanced Performance Metrics:** Includes usage percentage, offensive and defensive ratings and a custom value index.
- **Playmaking & Defense:** Covers assists, rebounds, steals, blocks and turnovers.

The table below provides **detailed descriptions** of each column to help you understand what they represent and how they can be used for analysis.

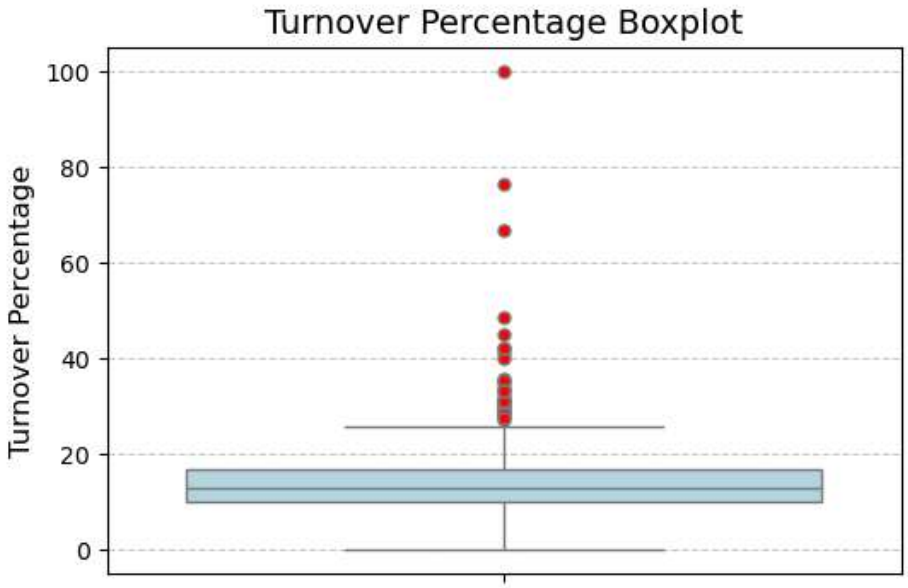
Column	Description
name	Player's full name (e.g. Lebron James)
team	The team the player is listed under in the dataset (e.g. 'Lal' is the Los Angeles Lakers)
current_team	Indicates whether the player is currently playing for the listed team (Yes/No)
position	Position on Court
age	Player's Age
games_played	Total number of games the player has played in the season
minutes_per_game	Average minutes the player plays per game
usage_percentage	The percentage of a team's offensive plays that involve the player while they are on the court. A higher value means the player is more involved in scoring opportunities
turnover_percentage	The percentage of a player's possessions that result in a turnover (losing the ball to the other team). Lower values are better
free_throw_attempts	The total number of free throws attempted by the player
free_throw_percentage	The player's success rate in free throws, expressed as a percentage (Free Throws Made ÷ Free Throws Attempted) × 100)
two_point_attempts	The number of two-point shots attempted (inside the three-point line)
two_point_percentage%	The success rate of two-point shots
three_point_attempts	The number of three-point shots attempted (beyond the three-point line)
three_point_percentage	The success rate of three-point shots
effective_field_goal_percentage	A weighted shooting percentage that adjusts for three-pointers being worth more than two-pointers
true_shooting_percentage%	A more advanced shooting metric that accounts for free throws and three-pointers to measure scoring efficiency
points_per_game	The average number of points a player scores per game
rebounds_per_game	The average number of rebounds (missed shots recovered) per game
assists_per_game	The average number of assists (passes leading to a basket) per game
steals_per_game	The average number of times the player takes the ball away from the opposing team per game
blocks_per_game	The average number of blocked shots per game
turnovers_per_game	The average number of times the player loses possession per game
points_plus_rebounds	The player's total points and rebounds combined per game
points_plus_assists	The player's total points and assists combined per game
points_rebounds_assists	The player's total points, rebounds, and assists per game
value_index	A custom metric evaluating the player's overall impact. This may be calculated from multiple stats, but the exact formula depends on the dataset provider
offensive_rating	An estimate of the player's offensive efficiency, measuring points produced per 100 possessions. Higher values indicate better offensive play
defensive_rating	An estimate of the player's defensive effectiveness, measuring points allowed per 100 possessions. Lower values indicate stronger defensive play

Exploratory Data Analysis

In [7]:
```
#A quick glance at the statistical summary has given me concern for some of the max values - time to inv
display(df[['turnover_percentage','free_throw_attempts','three_point_percentage']].describe())
```

	turnover_percentage	free_throw_attempts	three_point_percentage
count	616.000000	616.000000	616.000000
mean	13.896753	64.977273	0.288140
std	8.346379	82.088052	0.165355
min	0.000000	0.000000	0.000000
25%	9.975000	8.750000	0.232500
50%	13.000000	32.500000	0.332500
75%	16.750000	88.250000	0.377250
max	100.000000	537.000000	1.000000

In [8]:
```
#Bopxplot
plt.figure(figsize=(6, 4))
sns.boxplot(data=df['turnover_percentage'],
            color='lightblue',
            flierprops={'marker': 'o', 'markerfacecolor': 'red', 'markersize': 5})

plt.title('Turnover Percentage Boxplot', fontsize=14)
plt.ylabel('Turnover Percentage', fontsize=12)

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.show()
```



The majority of players have turnover percentages between 5% and 20%.

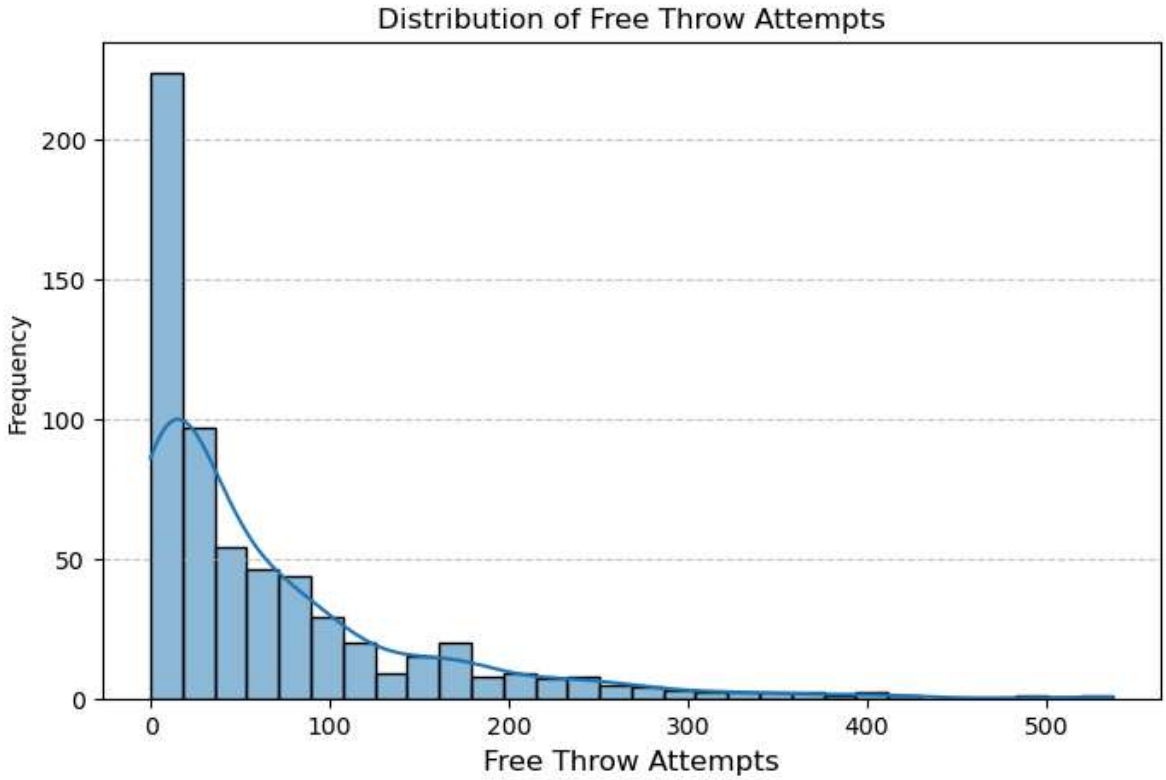
There is a long tail of ***extreme high turnover percentages***, meaning some players lose possession at an unusually high rate.

In [10]:
```
#Investigate who has a 100% turnover percentage
turnover_max = df.loc[df['turnover_percentage'].idxmax()]
display(turnover_max[['team', 'current_team', 'position', 'games_played', 'minutes_per_game', 'turnover_
```

name	team	current_team	position	games_played	minutes_per_game	turnover_percentage
Sidy Cissoko	San	No	G	17	3.2	10.0
Sidy Cissoko	Por	Yes	G	2	2.6	100.0

`Turnover Percentage Max = 100%`. This suggests Sidy Cissoko has lost the ball on every single possession since he was traded to Portland, which is likely why he has **only played 6 minutes** in the league this season.

In [68]:
```
plt.figure(figsize=(8, 5))

#Histogram
sns.histplot(df['free_throw_attempts'], bins=30, kde=True, edgecolor='black')

plt.xlabel('Free Throw Attempts', fontsize=12)
plt.ylabel('Frequency')
plt.title('Distribution of Free Throw Attempts')

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.show()
```



Positively Skewed Distribution: Most players take very few free throw attempts, with a long tail extending toward higher values and potentially superstar players who draw a lot of fouls.

Majority of Players: The highest frequency is concentrated below **50 free throw attempts**, meaning most players are not getting to the free throw line more than once a game.

Small Group of High-Volume Shooters: A few players attempt over **200-500 free throws**, indicating ***elite scorers*** who draw a lot of fouls.

Outliers: The extreme values at the far right suggest a few players dominate in free throw attempts, potentially ***star players*** or players on a ***'bad' team*** where the rest of the team rely on them to score as many points as they can every game.

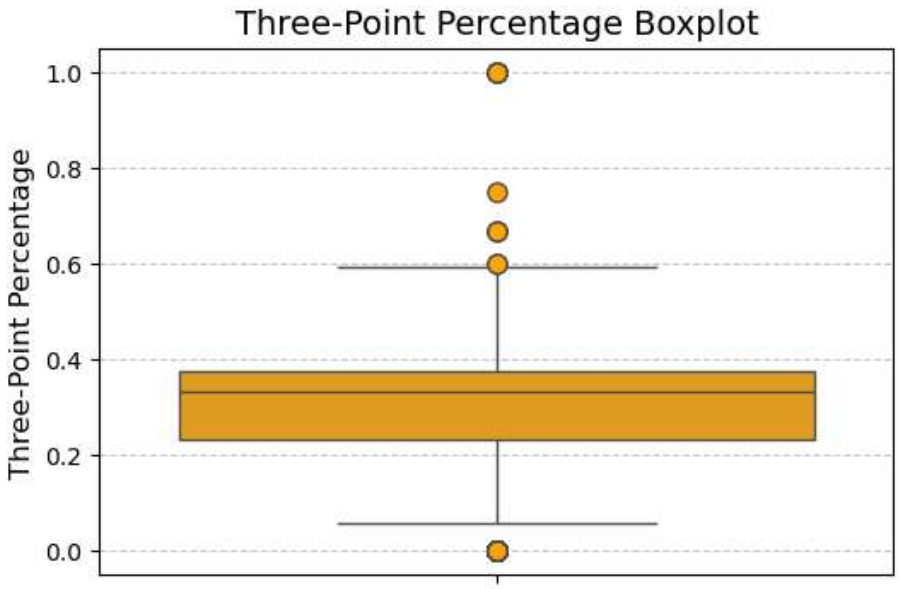
In [14]:
```
#Find out who is the max free throw attempter
free_throw_atts = df.loc[[df['free_throw_attempts'].idxmax()]]
display(free_throw_atts[['team', 'current_team', 'position', 'age', 'minutes_per_game','free_throw_attem
```

name	team	current_team	position	age	minutes_per_game	free_throw_attempts	free_throw_percentage
Shai Gilgeous-Alexander	Okc	Yes	G	26.6	34.2	537	0.896

Free Throw Attempts Max: 537

After investigating, **Shai** is having an ***incredible season***, and **9 free throws per game** isn't a ***ludicrous average*** at all.

In [16]:
```
plt.figure(figsize=(6, 4))

sns.boxplot(data=df['three_point_percentage'],
            color='orange',
           flierprops={'marker': 'o', 'markerfacecolor': 'orange', 'markersize': 8, 'linestyle': 'none'}
)

plt.ylabel('Three-Point Percentage', fontsize=12)
plt.title('Three-Point Percentage Boxplot', fontsize=14)
plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.show()
```



Most players fall within a three point shooting range (20-45%).

Outliers at **100% accuracy** could indicate players who took very few attempts but made all their shots.

Low outliers near **0%** might suggest players who are ***poor shooters*** but still attempt threes.

In [18]:
```
three_point_pctmax = df.loc[[df['three_point_percentage'].idxmax()]]
display(three_point_pctmax[['team', 'current_team', 'position', 'age', 'three_point_attempts','three_poi
```

name	team	current_team	position	age	three_point_attempts	three_point_percentage
Anthony Davis	Dal	Yes	F	32.0	2	1.000
Anthony Davis	Lal	No	F	32.0	94	0.298

Three Point Percentage Max: 100%

Due to the **small sample** of **2 attempts**, it is possible that **Anthony Davis** made both.

In [20]:
```
three_point_pctmin = df.loc[[df['three_point_percentage'].idxmin()]]
display(three_point_pctmin[['team', 'current_team', 'position', 'age', 'three_point_attempts','three_poi
```

name	team	current_team	position	age	three_point_attempts	three_point_percentage
Kai Jones	Dal	Yes	C	24.1	0	0.0
Kai Jones	Lac	No	C	24.1	2	0.0

Confirmed – **Kai Jones** is a **C (Centre)**, and they are certainly not notorious for their **three-point shooting.**

He has ***missed both of his attempts*** before getting traded.

Feature Engineering

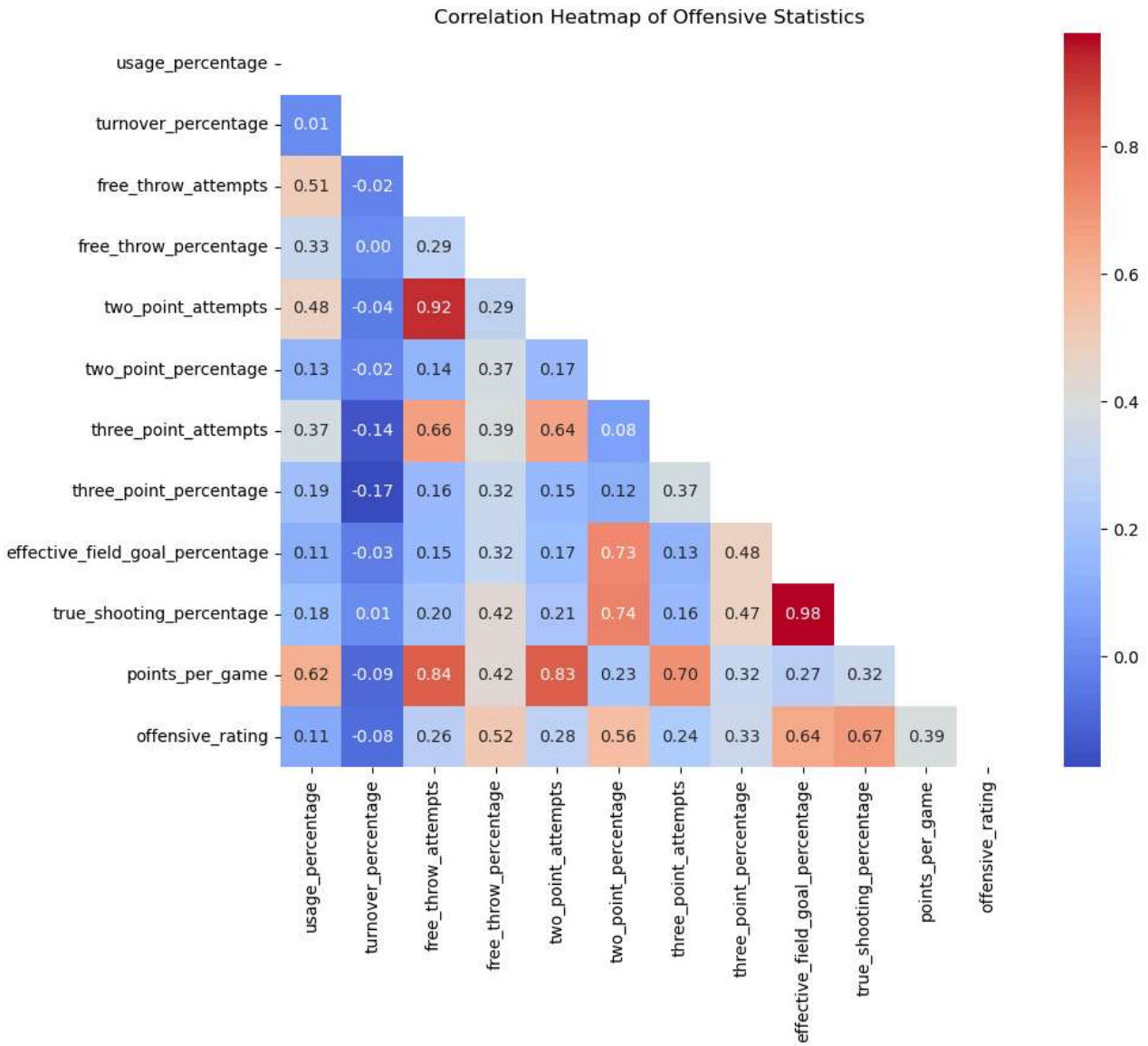
Why Analyse Offensive Correlations?

To determine the most important factors contributing to **offensive efficiency**, I will analyse the relationships between various offensive statistics. Understanding how these metrics interact allows me to:

- **Identify the most impactful offensive attributes** – which stats correlate strongly with overall scoring and efficiency?
- **Discover my weighted efficiency formula** – giving more weight to key factors that have the greatest influence on an offensive player.

By focusing on **correlation**, I ensure that my final **Offensive Efficiency Model** is built on ***data-driven insights*** rather than arbitrary weighting.

In [23]:
```
#Selected only offensive statistics
offensive_stats = [
    'usage_percentage', 'turnover_percentage', 'free_throw_attempts', 'free_throw_percentage',
    'two_point_attempts', 'two_point_percentage', 'three_point_attempts', 'three_point_percentage',
    'effective_field_goal_percentage', 'true_shooting_percentage', 'points_per_game',
    'offensive_rating'
]

corr = df[offensive_stats].corr()

#Create mask for upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool))

#Plot heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr, mask=mask, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap of Offensive Statistics")
plt.show()
```



Key Observations from the Correlation Heatmap:

Strongest Correlations with Points Per Game:

- **Two-Point Attempts** (**0.84**)
- **Three-Point Attempts** (**0.64**)
- **True Shooting %** (**0.70**)
- **Effective Field Goal %** (**0.47**)

Strongest Correlations with Offensive Rating:

- **True Shooting %** (**0.67**)
- **Effective Field Goal %** (**0.64**)
- **Two-Point Attempts** (**0.52**)

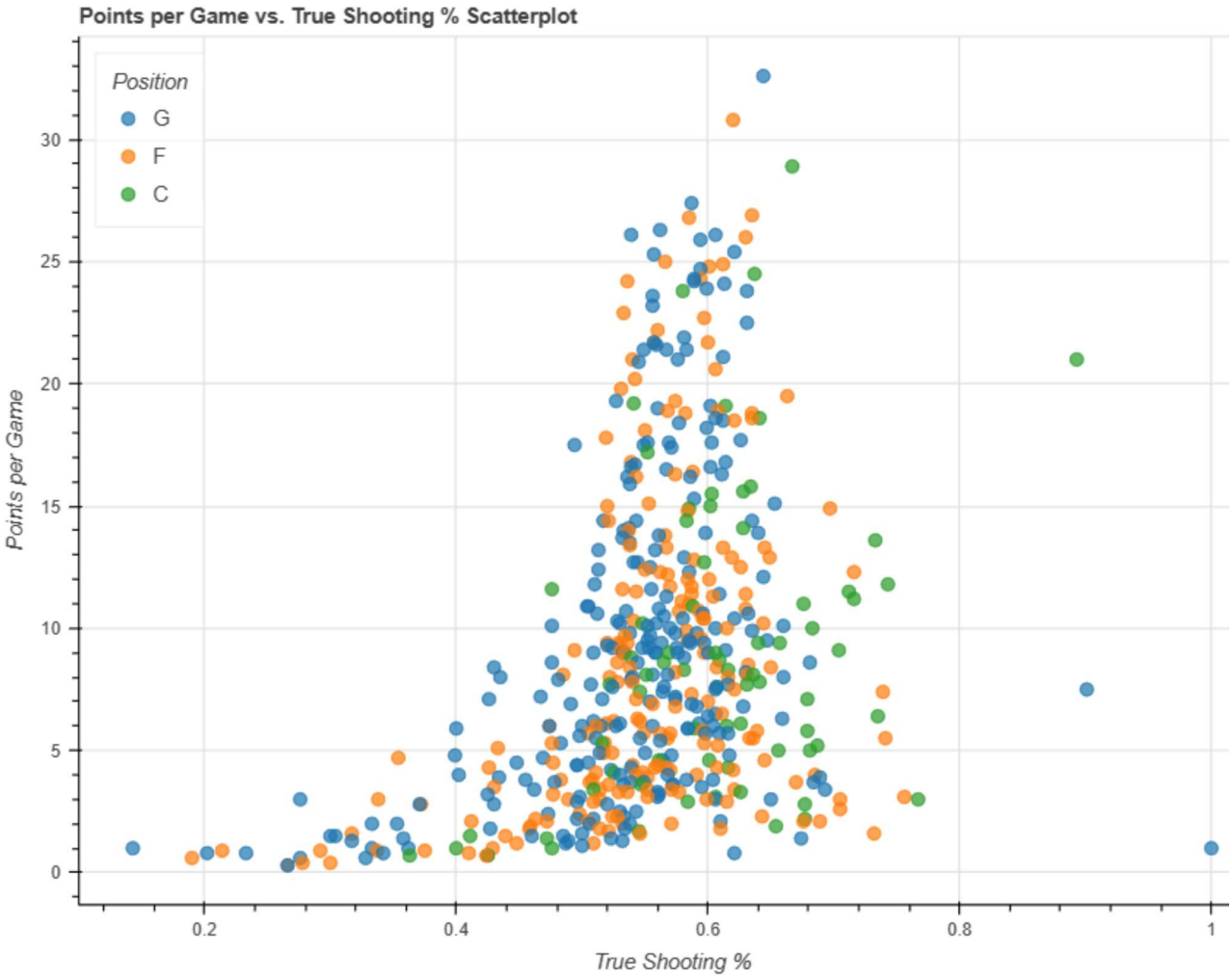
Metrics to Drop or Give Lower Weight:

- **Turnover %** has a **low correlation** with both **points_per_game** and **offensive_rating**.
- **Free throw attempts and percentage** have **lower impact** compared to other **shooting efficiency metrics**.

In [25]:
```
output_notebook()
```

```
x_metric = 'true_shooting_percentage'
y_metric = 'points_per_game'

#Filter dataset for players currently on their team and valid TS%
scat_df = df[(df['current_team'] == 'Yes') & (df['true_shooting_percentage'] > 0) & (df['true_shooting_p

positions = scat_df['position'].unique().tolist()

source = ColumnDataSource(scat_df)

#Create figure
fig = figure(
    title='Points per Game vs. True Shooting % Scatterplot',
    x_axis_label='True Shooting %',
    y_axis_label='Points per Game',
    width=800, height=600,
    tools='pan,box_zoom,reset,save'
)

#Prevent auto-rendering
dummy_var = fig

#Scatter plot with colour mapping by position
scatter = fig.scatter(
    x=x_metric, y=y_metric, source=source,
    size=8, alpha=0.7,
    color=factor_cmap('position', palette=Category10[len(positions)], factors=positions),
    legend_field='position'
)

#Add HoverTool
hover = HoverTool(
    tooltips=[
        ('Player', '@name'),
        ('Team', '@team'),
        ('Position', '@position'),
        ('True Shooting %', '@true_shooting_percentage'),
        ('Points per Game Average', '@points_per_game')
        ],
    renderers=[scatter]
)
fig.add_tools(hover)

#Configure legend
fig.legend.title = 'Position'
fig.legend.location = 'top_left'

show(fig, notebook_handle=True)
push_notebook() #Stops duplicate scatterplots from appearing
```

BokehJS 3.6.0 successfully loaded.



Points per Game vs. True Shooting % – Scatter Plot Analysis

Understanding the Chart

This scatter plot visualises the relationship between **Points per Game (PPG)** and **True Shooting Percentage (TS%)**, with each point representing an individual player. The data is colour-coded by position:

- **Guards (G) – Orange**
- **Forwards (F) – Blue**
- **Centres (C) – Green**

Key Observations

Positive Correlation Between True Shooting % and Points per Game

- Players with a **higher TS% generally score more points per game**, reinforcing the idea that efficient shooters tend to be high-volume scorers.
- The majority of players cluster between **0.45 and 0.65 TS%**, with PPG ranging between **0 and 15.**

High-Scoring Outliers

- Shai and Giannis exceed **30+ PPG**, showing elite offensive performers with exceptional efficiency.

Positional Trends

- **Guards (Orange) and Forwards (Blue)** dominate the **high-scoring** category, suggesting they take more offensive responsibilities.
- **Centres (Green)** generally have **lower scoring totals but higher efficiency**, likely due to scoring closer to the basket with high-percentage shots.
- The spread of **TS% is wider for Guards**, indicating variation in shot selection (some focus on three-pointers while others attack the rim more aggressively).

This scatter plot **confirms the expected relationship** between scoring and efficiency while also revealing **positional trends** in shot selection.

Metrics & Proposed Weighting for Offensive Efficiency

1. True Shooting % (0.35 weight)

- **Why?** This is the **strongest** indicator of offensive efficiency, with a **0.98 correlation** with points per game.
- **What it measures:** Accounts for all types of scoring (field goals, three-pointers, and free throws) in a single efficiency metric.
- **Why it's weighted highest:** It **directly predicts** a player's scoring output while adjusting for shot type.

2. Effective Field Goal % (0.25 weight)

- **Why?** Has a strong **0.73 correlation** with true shooting % and a **0.64 correlation** with offensive rating.
- **What it measures:** Adjusts shooting percentage by giving **extra weight to three-pointers**, since they are worth more than two-point shots.
- **Why it's important:** Unlike raw field goal percentage, **eFG% accounts for shot value**, making it a more precise measure of shooting efficiency.

3. Two-Point Attempts (0.20 weight)

- **Why?** Has an **extremely high correlation (0.84) with points per game** and a **0.52 correlation with offensive rating**.
- **What it measures:** Frequency of **inside-the-arc** shot attempts.
- **Why it's important:** Players who take a lot of two-point shots often **score more overall**, especially big men and mid-range specialists.

4. Three-Point Attempts (0.15 weight)

- **Why?** Has a **0.64 correlation** with points per game and a **0.39 correlation** with offensive rating.
- **What it measures:** Volume of **three-point shot attempts**.
- **Why it's weighted lower than twos:** While important, three-pointers are taken at **lower efficiency** than two-pointers, and volume alone does not always equate to scoring efficiency.

5. Usage % (0.05 weight)

- **Why?** Has a **0.62 correlation** with points per game, but only a **0.11 correlation with offensive rating**.
- **What it measures:** How often a player is involved in offensive plays.
- **Why it's weighted lowest:** High usage doesn't always mean **high efficiency** - some players take a lot of shots but are not efficient scorers.

Why These Weightings?

This weighting system prioritises **shooting efficiency over raw volume** because:

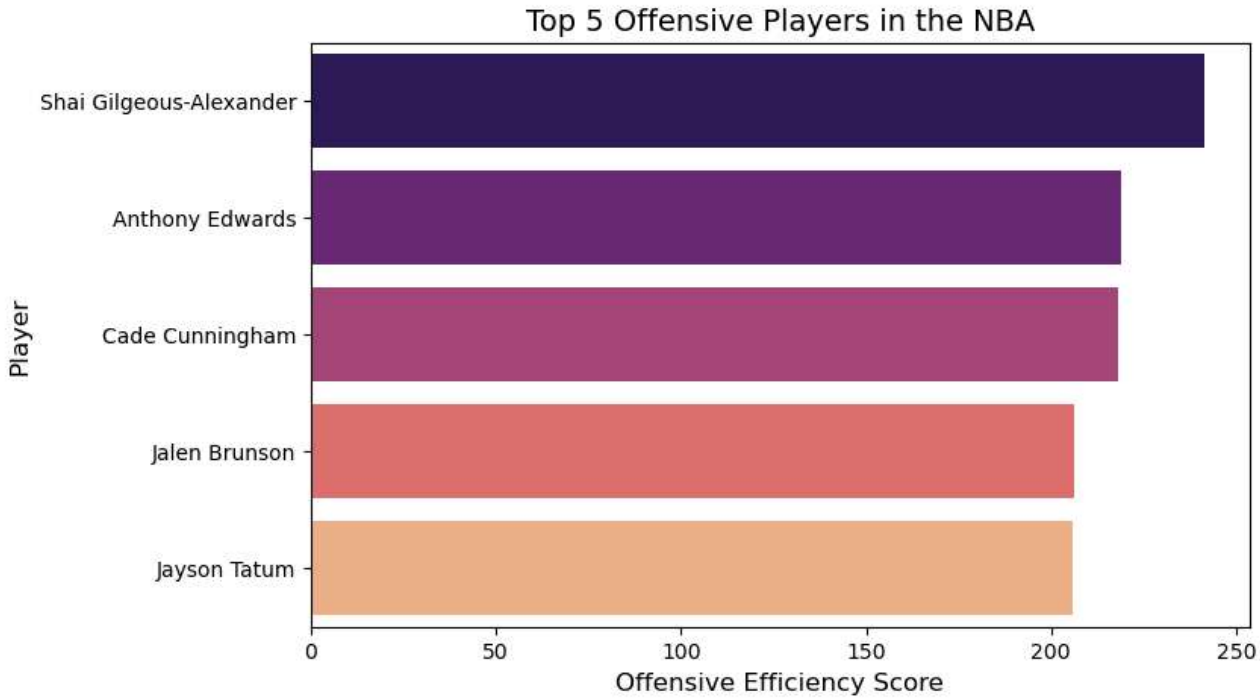
- **True Shooting % and Effective Field Goal %** are the **best indicators** of scoring efficiency.
- **Two-Point and Three-Point Attempts** measure **shot volume**, which correlates with scoring but does not guarantee efficiency.
- **Usage %** has some value but is **less predictive** of overall offensive rating.

By structuring the **Offensive Efficiency Model** around these correlations, I ensure it is **data-driven and reflects real performance trends** rather than arbitrary weightings.

In [28]:
```
#Create new offensive efficiency metric with weighted contributions
off_efficiency_metrics = df[['team', 'position', 'games_played','true_shooting_percentage', 'effective_f
                             'two_point_attempts', 'three_point_attempts', 'usage_percentage']].copy()

#Apply weightings
off_efficiency_metrics['offensive_efficiency'] = (
    off_efficiency_metrics['true_shooting_percentage'] * 0.35 +
    off_efficiency_metrics['effective_field_goal_percentage'] * 0.25 +
    off_efficiency_metrics['two_point_attempts'] * 0.20 +
    off_efficiency_metrics['three_point_attempts'] * 0.15 +
    off_efficiency_metrics['usage_percentage'] * 0.05
)

#Sort data frame to show the 5 most offensively effecient players in the NBA
sort_off_eff = off_efficiency_metrics.sort_values(by='offensive_efficiency', ascending=False)
display(sort_off_eff[['position', 'true_shooting_percentage', 'effective_field_goal_percentage',
                             'two_point_attempts', 'three_point_attempts', 'usage_percentage','offensive
```

name	position	true_shooting_percentage	effective_field_goal_percentage	two_point_attempts	three_point_attempts
Shai Gilgeous-Alexander	G	0.644	0.575	941	342
Anthony Edwards	G	0.587	0.538	630	607
Cade Cunningham	G	0.557	0.512	814	357
Jalen Brunson	G	0.606	0.553	754	358
Jayson Tatum	F	0.585	0.541	582	585

In [29]:
```
plt.figure(figsize=(8, 5))

#Plot a horizontal bar chart
sns.barplot(data=sort_off_eff.head(5),
            x='offensive_efficiency',
            y=sort_off_eff.head(5).index,
            palette='magma')


plt.xlabel('Offensive Efficiency Score', fontsize=12)
plt.ylabel('Player', fontsize=12)
plt.title('Top 5 Offensive Players in the NBA', fontsize=14)

plt.show()
```



Top 5 Offensive Players - Key Takeaways

Shai Gilgeous-Alexander (OKC) - Clear Leader

- **True Shooting%:** 0.644 (Best in the group)
- **Usage:** 34.5% (Highest)
- **Efficiency Score:** 241.59 (Well ahead of others)
  SGA is both **highly efficient** and **heavily relied upon**.

Anthony Edwards (MIN) & Cade Cunningham (DET) - Volume Scorers

- **Usage:** Edwards 31.7%, Cunningham 32.6%
- **Scoring Styles:**
  - **Cunningham** - 814 two-point attempts
  - **Edwards** - 607 three-point attempts
- **Efficiency Scores:** Edwards 218.97, Cunningham 218.30
  Both dominate possessions but with **different shot distributions**.

Jalen Brunson (NYK) - Efficient Despite Lower Usage

- **True Shooting%:** 0.606 (2nd best)
- **Usage:** 29.5% (Lower than others)
- **Efficiency Score:** 206.33
  Highly **selective and efficient**, making the most of his opportunities.

Jayson Tatum (BOS) - Balanced Scorer

- **Two-Point Attempts:** 582
- **Three-Point Attempts:** 585
- **Usage:** 30.9%
- **Efficiency Score:** 206.03

  A **versatile** offensive player contributing inside and outside the three point line.

Summary:

🔝 **Gilgeous-Alexander** is the ***most efficient*** and ***dominant***.

⛹️ **Edwards** and **Cunningham** score in volume but ***differ in shot selection***.

👍🏾 **Brunson** ***maximises efficiency*** with fewer shots.

⭕ **Tatum** offers a ***well-rounded offensive game***.

Why Analyse Defensive Metrics?

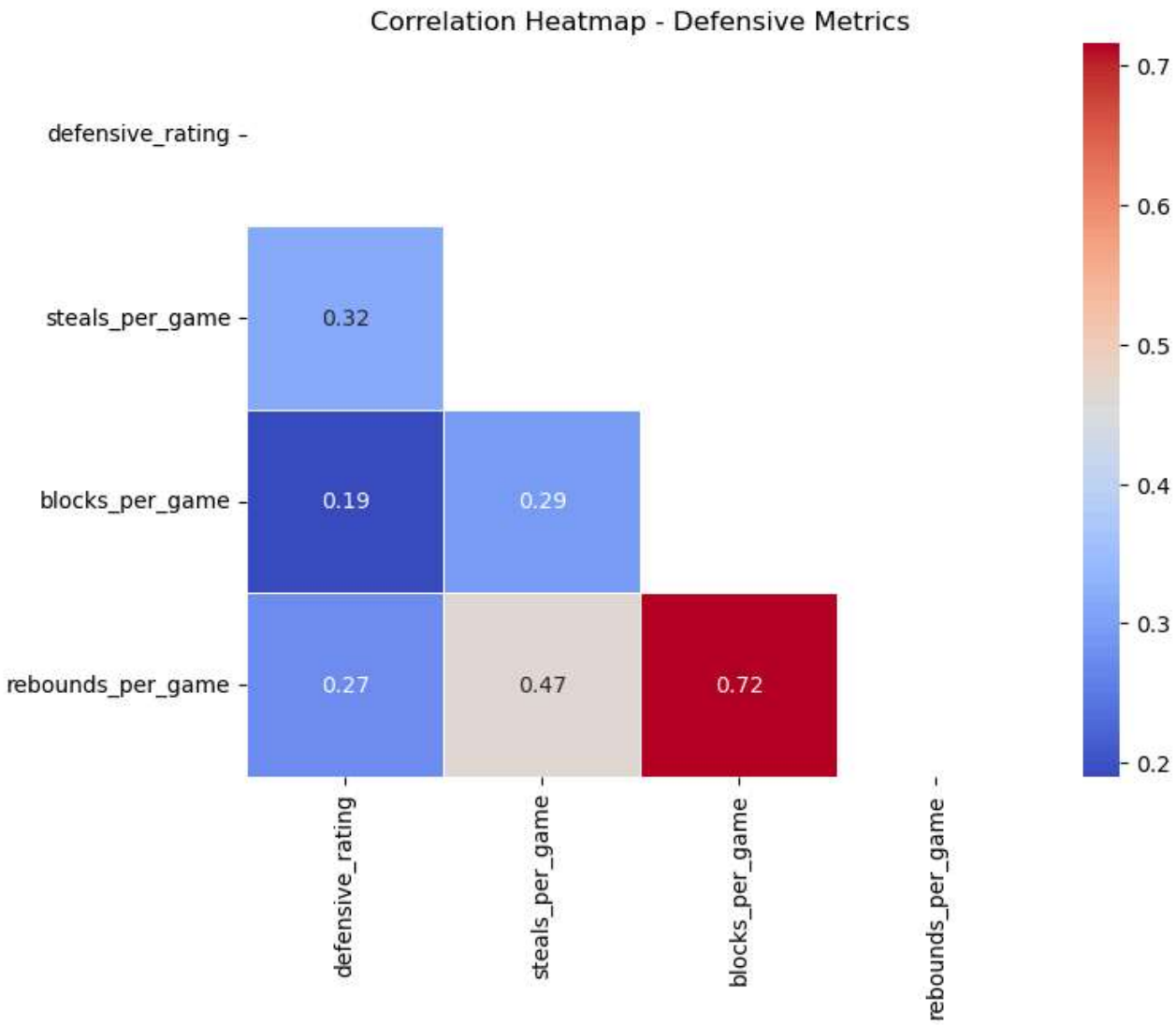
To determine the key factors influencing **defensive performance**, I will analyse the correlations between **defensive rating** and other defensive statistics.

Why This Matters:

- **Identify key defensive contributors** – Which metrics have the strongest impact on defensive success?
- **Optimise defensive efficiency calculations** – Helps assign the correct weightings in my model.
- **Compare defensive playstyles** – Understanding differences between **rim protectors** (shot-blocking big men) and **perimeter defenders** (lockdown guards & wings).

By focusing on these correlations, I ensure my **Defensive Efficiency Model** is built on **data-driven insights**, not assumptions.

In [32]:
```
#Select only defensive-related columns
defensive_metrics = ['defensive_rating', 'steals_per_game', 'blocks_per_game', 'rebounds_per_game']

#Compute correlation
corr_def = df[defensive_metrics].corr()

#Create mask to show only lower triangle
mask = np.triu(np.ones_like(corr_def, dtype=bool))

#Plot heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(corr_def, mask=mask, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title("Correlation Heatmap - Defensive Metrics")
plt.show()
```



Choosing Defensive Metrics

Based on the **correlation heatmap**, I can determine the most important defensive stats that influence **Defensive Rating**.

Findings:

- **Steals per game (0.32 correlation)** – A strong indicator of **defensive pressure on the perimeter**.
  ➡️**Contributes 35%** to defensive efficiency.
- **Blocks per game (0.19 correlation)** – Has **less impact than steals**, but remains crucial for **rim protection**.
  ➡️**Contributes 20%** to defensive efficiency.
- **Rebounds per game (0.27 correlation)** – Shows a **strong relationship with blocks (0.72 correlation)**, emphasising the role of **rebounding in defensive success**.
  ➡️**Contributes 30%** to defensive efficiency.
- **Defensive Rating (Baseline)** – Including this metric ensures that the model **captures overall defensive impact**.
  ➡️**Contributes 15%** to defensive efficiency.

By weighting these metrics appropriately, I ensure that my **Defensive Efficiency Model** reflects **real defensive contributions** rather than arbitrary assumptions.

In [34]:
```
#Defensive weightings based on correlation heatmap
df['steals_weighted'] = df['steals_per_game'] * 0.35
```

```
df['rebounds_weighted'] = df['rebounds_per_game'] * 0.30
df['blocks_weighted'] = df['blocks_per_game'] * 0.20
df['def_rating_weighted'] = df['defensive_rating'] * 0.15

#Defensive efficiency score
df['defensive_efficiency'] = (
    df['steals_weighted'] + df['rebounds_weighted'] +
    df['blocks_weighted'] + df['def_rating_weighted']
)
#Filter players who have played at least 20 games
df_20games = df[df['games_played'] >= 20]

#Sort and display top 5 defensive players
top_defensive_players = df_20games[
    ['team', 'position', 'games_played', 'steals_per_game',
     'blocks_per_game', 'rebounds_per_game','defensive_rating', 'defensive_efficiency']
].sort_values(by='defensive_efficiency', ascending=False).head(5)

display(top_defensive_players)
```

name	team	position	games_played	steals_per_game	blocks_per_game	rebounds_per_game	defensive_rating	def
Walker Kessler	Uta	C	47	0.6	2.3	12.1	112.1	
Nikola Jokic	Den	C	55	1.8	0.7	12.7	107.3	
Victor Wembanyama	San	F	46	1.1	3.8	11.0	107.2	
Jalen Johnson	Atl	F	36	1.6	1.0	10.0	111.1	
Anthony Davis	Lal	F	42	1.3	2.1	11.9	106.2	

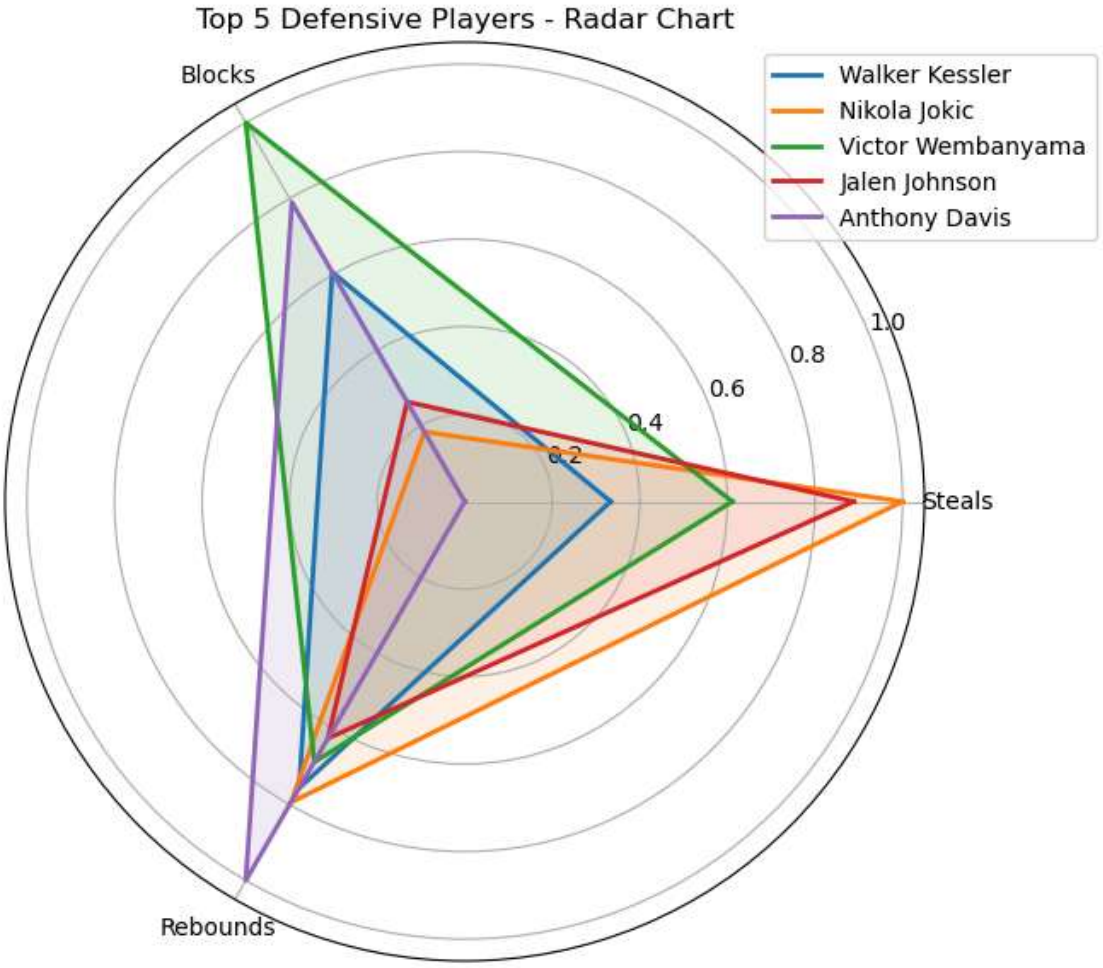
Why Normalise the Data?

Defensive stats like **steals, blocks, and rebounds** have different scales, which can make one stat appear more important than another.

Normalisation adjusts all values to a **0-1 range**, ensuring a fair comparison and making the radar chart more balanced and readable.

In [36]:
```
#Define players and key defensive metrics
players = top_defensive_players.index.tolist()
metrics = ['Steals', 'Blocks', 'Rebounds']

#Extract relevant data
data = df.loc[players, ['steals_per_game', 'blocks_per_game', 'rebounds_per_game']].values

#Normalise data for better scaling
data = data / data.max(axis=0)

#Radar Chart Setup
angles = np.linspace(0, 2 * np.pi, len(metrics), endpoint=False).tolist()
angles += angles[:1]  # Complete the loop

fig, ax = plt.subplots(figsize=(7, 7), subplot_kw=dict(polar=True))

#Plot each player's defensive metrics
for i, player in enumerate(players):
    values = data[i].tolist()
    values += values[:1]  # Repeat first value to close the circle
    ax.plot(angles, values, label=player, linewidth=2)
    ax.fill(angles, values, alpha=0.1)
```

```
#Format the chart
ax.set_xticks(angles[:-1])
ax.set_xticklabels(metrics)
ax.set_title('Top 5 Defensive Players - Radar Chart')
ax.legend(loc='upper right', bbox_to_anchor=(1.2, 1))

plt.show()
```



Top 5 Defensive Players - Key Takeaways

Walker Kessler (UTA) - Defensive Anchor

- **Rebounding:** 12.1 rebounds per game (2nd highest in top 5).
- **Shot Blocking:** 2.3 blocks per game, strong rim protection.
- Excelling in protecting the paint.

Nikola Jokic (DEN) - All-Around Defensive Presence

- **Steals & Blocks:** 1.8 steals per game (highest in top 5) & 0.7 blocks per game.
- **Rebounding:** 12.7 rebounds per game, making him an elite defensive rebounder.
- Versatile defender with strong anticipation and rebounding impact.

Victor Wembanyama (SAS) - Rim Protection Specialist

- **Blocks:** 3.8 blocks per game (by far the most in top 5).
- **Rebounding:** 11.0 rebounds per game, controlling the boards.
- The most dominant shot-blocker.

Jalen Johnson (ATL) - Defensive Playmaker

- **Steals:** 1.6 steals per game (2nd highest in top 5).
- **Blocks & Rebounds:** 1.0 blocks and 10.0 rebounds per game.
- A well-rounded defender who disrupts passing lanes and controls the glass.

Anthony Davis (LAL) - Defensive Veteran

- **Steals & Blocks:** 1.3 steals and 2.1 blocks per game.
- **Rebounding:** 11.9 rebounds per game, a key presence inside.
- An elite two-way player known for his defensive versatility and rim protection.

Summary:

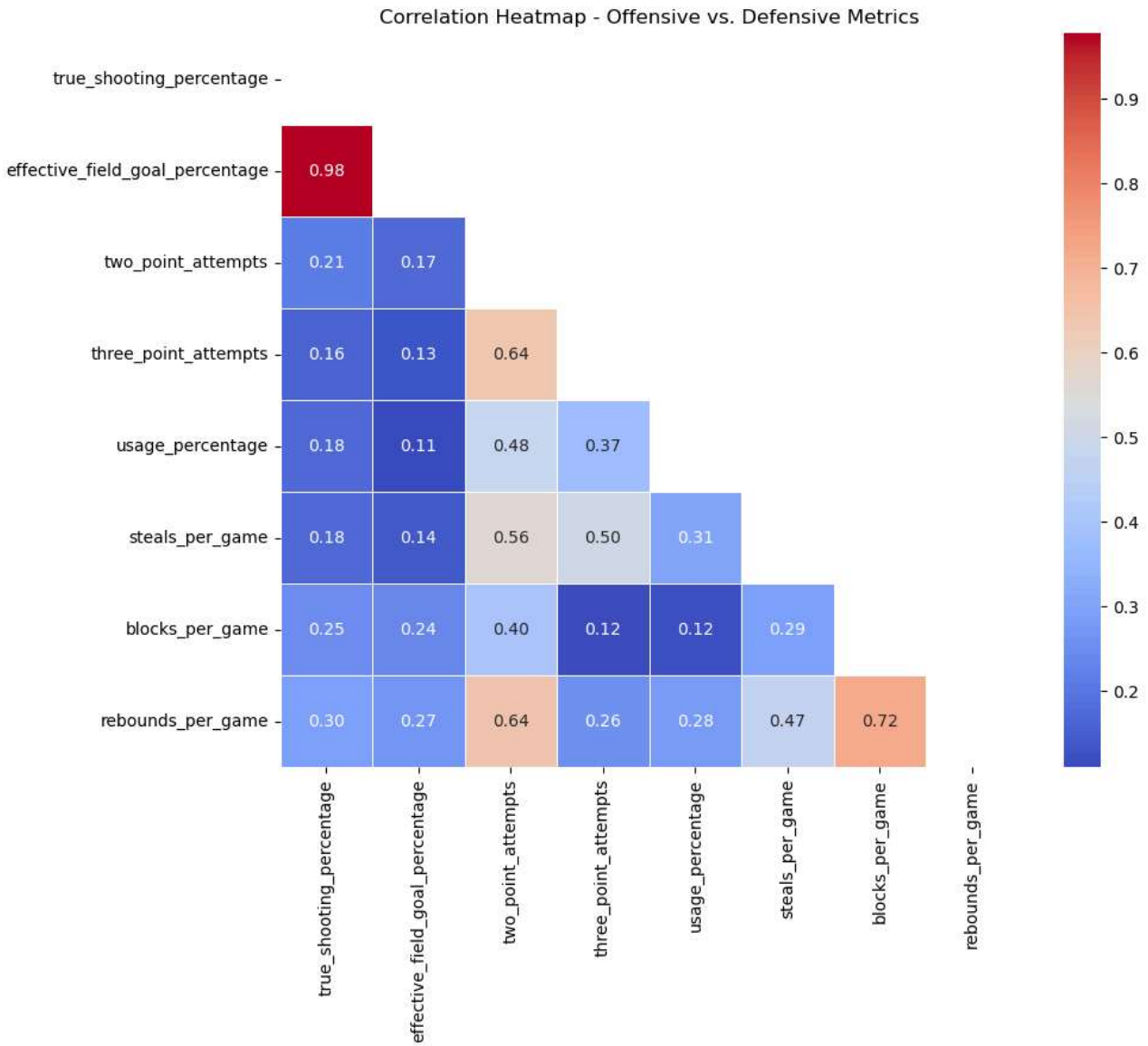
💪 Kessler & Jokic **dominate** the boards.

⛔ Wembanyama is the **best shot-blocker.**

🏆 Johnson & Davis bring ***defensive versatility*** with steals, blocks and rebounding.

🔝 Who truly are the very best in the NBA?

In [39]:
```
#Selecting relevant offensive and defensive metrics for correlation analysis
off_def_corr_metrics = df[['true_shooting_percentage',
                           'effective_field_goal_percentage',
                           'two_point_attempts',
                           'three_point_attempts',
                           'usage_percentage',
                           'steals_per_game',
                           'blocks_per_game',
                           'rebounds_per_game'
                          ]]

#Compute correlation matrix
corr_matrix = off_def_corr_metrics.corr()

# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))

#Create the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, mask=mask, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title("Correlation Heatmap - Offensive vs. Defensive Metrics")
plt.show()
```



Key Findings from Correlation Analysis

- **Strong Correlation Between True Shooting % and Effective Field Goal % (0.98)**
  - Since these metrics are nearly identical in their impact, I only need **one** in my overall ranking to **avoid redundancy.**
- **Low Correlation Between Offensive and Defensive Metrics** (As Expected)
  - **Offensive stats** like **Effective Field Goal Percentage (eFG%)** and **Usage Percentage** have little to no correlation with **defensive stats** such as **Steals, Blocks, or Rebounds.**
  - This suggests that **being a great scorer doesn't necessarily translate into being a great defender.**

Implications for Final Rankings

- I'll need to **carefully balance weightings** in my final model to ensure **both offensive and defensive contributions are properly represented.**
- Selecting **complementary metrics** (rather than redundant ones) will improve the accuracy of player evaluations.

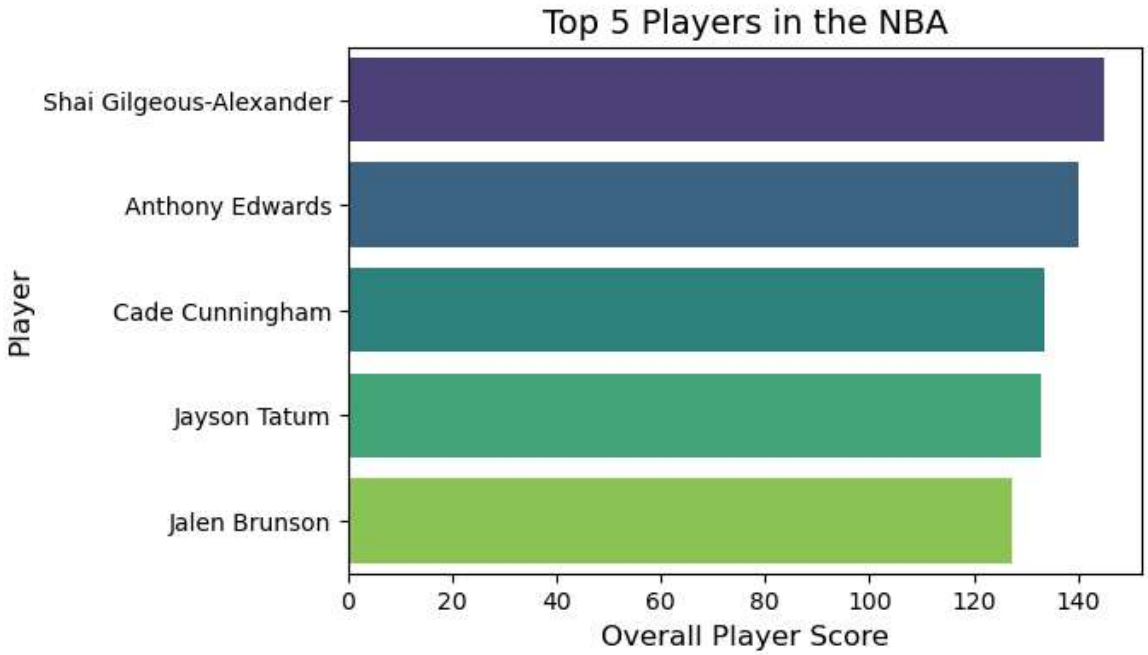
In [41]:
```
#Create new dataframe for balanced approach
df_all_around = df.copy()
```

```
#Offensive Contributions (50% Total)
df_all_around['ts_weighted'] = df_all_around['true_shooting_percentage'] * 0.15
df_all_around['usage_weighted'] = df_all_around['usage_percentage'] * 0.10
df_all_around['three_pt_weighted'] = df_all_around['three_point_attempts'] * 0.10
df_all_around['two_pt_weighted'] = df_all_around['two_point_attempts'] * 0.10
df_all_around['points_weighted'] = df_all_around['points_per_game'] * 0.05

#Defensive Contributions (50% Total)
df_all_around['steals_weighted'] = df_all_around['steals_per_game'] * 0.15
df_all_around['blocks_weighted'] = df_all_around['blocks_per_game'] * 0.15
df_all_around['rebounds_weighted'] = df_all_around['rebounds_per_game'] * 0.10
df_all_around['def_rating_weighted'] = df_all_around['defensive_rating'] * 0.10

#Overall Player Score
df_all_around['overall_player_score'] = (
    df_all_around['ts_weighted'] + df_all_around['usage_weighted'] + df_all_around['three_pt_weighted']
    df_all_around['two_pt_weighted'] + df_all_around['points_weighted'] + df_all_around['steals_weighted
    df_all_around['blocks_weighted'] + df_all_around['rebounds_weighted'] + df_all_around['def_rating_we
)

#Sort and display the top 5 all-around players
top_all_around_players = df_all_around[['team', 'position', 'games_played', 'overall_player_score']]\
    .sort_values(by='overall_player_score', ascending=False).head(5)

display(top_all_around_players)
```

name	team	position	games_played	overall_player_score
Shai Gilgeous-Alexander	Okc	G	60	144.95660
Anthony Edwards	Min	G	60	140.15805
Cade Cunningham	Det	G	57	133.42855
Jayson Tatum	Bos	F	58	132.87775
Jalen Brunson	Nyk	G	60	127.42090

In [42]:

```
#Set figure size
plt.figure(figsize=(6, 4))

#Plot a horizontal bar chart for overall player scores
sns.barplot(
    data=top_all_around_players,
    x='overall_player_score',
    y=top_all_around_players.index,
    palette='viridis'
)

#Labels and title
plt.xlabel('Overall Player Score', fontsize=12)
plt.ylabel('Player', fontsize=12)
plt.title('Top 5 Players in the NBA', fontsize=14)

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



Conclusion

The overall **top 5 players** are **identical to the offensive top 5**, reinforcing the idea that the **NBA is an offense-driven league**. A team's success is largely dictated by its ability to **score efficiently**, making **offensive metrics a major factor** in determining overall player impact.

While **defense remains crucial**, the data suggests that **elite overall players** are often those who **excel offensively**. This indicates that when teams look to **acquire or build around a star player**, **offensive ability is the primary factor** in decision-making.