EPL 2024/25 Analysis

In [231…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from mplsoccer import Radar, FontManager, add_image
```

In [232…
```
df = pd.read_csv("/Users/ashan/Desktop/Projects/Football-Analysis/Footbal
```

In [233…
```
df.shape
```

Out[233… (20, 148)

In [234…
```
unique_teams = df['Squad'].unique()
print(unique_teams)
```

```
['Arsenal' 'Aston Villa' 'Bournemouth' 'Brentford' 'Brighton' 'Chelsea'
 'Crystal Palace' 'Everton' 'Fulham' 'Ipswich Town' 'Leicester City'
 'Liverpool' 'Manchester City' 'Manchester Utd' 'Newcastle Utd'
 'Nottingham Forest' 'Southampton' 'Tottenham' 'West Ham' 'Wolves']
```

Data Preparation

Map teams and change the string values into numerical values with the numbers being based on their finishing league position

In [235…
```
team_mapping = {'Arsenal': 2, 'Aston Villa': 6,
                'Bournemouth': 9, 'Brentford': 10,
                'Brighton': 8, 'Chelsea': 4,
                'Crystal Palace': 12, 'Everton': 13,
                'Fulham': 11, 'Ipswich Town': 19,
                'Leicester City': 18, 'Liverpool': 1,
                'Manchester City': 3, 'Manchester Utd': 15,
                'Newcastle Utd': 5, 'Nottingham Forest': 7,
                'Southampton': 20, 'Tottenham': 17,
                'West Ham': 14, 'Wolves': 16
               }
```

In [236…
```
df['Team_Pos'] = df['Squad'].map(team_mapping)

print(np.sort(df['Team_Pos'].unique()))
```

```
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20]
```

In [237…
```
teams_by_pos = df.groupby('Team_Pos')['Squad'].unique()
print(teams_by_pos)
```

```
Team_Pos
1              [Liverpool]
2                [Arsenal]
3        [Manchester City]
4                [Chelsea]
5          [Newcastle Utd]
6            [Aston Villa]
7      [Nottingham Forest]
8                [Brighton]
9            [Bournemouth]
10             [Brentford]
11                [Fulham]
12       [Crystal Palace]
13               [Everton]
14             [West Ham]
15       [Manchester Utd]
16                [Wolves]
17             [Tottenham]
18       [Leicester City]
19         [Ipswich Town]
20           [Southampton]
Name: Squad, dtype: object
```

In [238…
```
Points_map = {
    'Liverpool': 84,
    'Arsenal': 74,
    'Manchester City': 71,
    'Chelsea': 69,
    'Newcastle Utd': 66,
    'Aston Villa': 66,
    'Nottingham Forest': 65,
    'Brighton': 61,
    'Bournemouth': 56,
    'Brentford': 56,
    'Fulham': 54,
    'Crystal Palace': 53,
    'Everton': 48,
    'West Ham': 43,
    'Manchester Utd': 42,
    'Wolves': 42,
    'Tottenham': 38,
    'Leicester City': 25,
    'Ipswich Town': 22,
    'Southampton': 12
}
```

In [239…
```
df['Points'] = df['Squad'].map(Points_map)
```

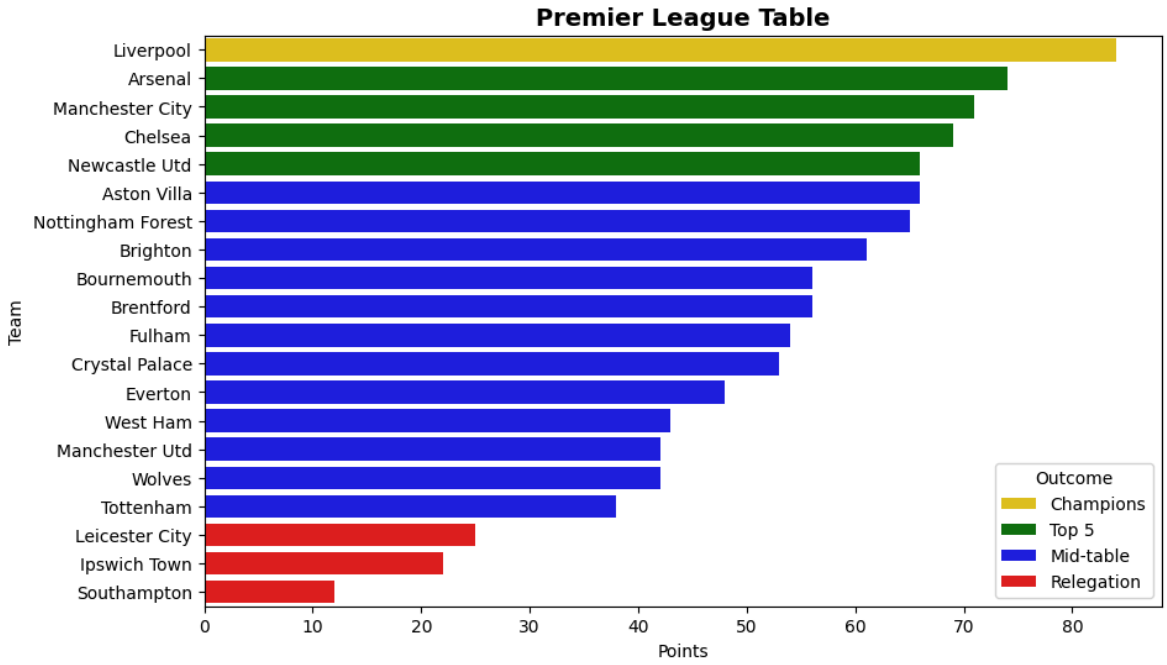
In [240…
```
League_Table = df[['Team_Pos', 'Squad', 'Points']].sort_values(by='Team_P
print(League_Table.set_index('Team_Pos'))
```

Team_Pos	Squad	Points
1	Liverpool	84
2	Arsenal	74
3	Manchester City	71
4	Chelsea	69
5	Newcastle Utd	66
6	Aston Villa	66
7	Nottingham Forest	65
8	Brighton	61
9	Bournemouth	56
10	Brentford	56
11	Fulham	54
12	Crystal Palace	53
13	Everton	48
14	West Ham	43
15	Manchester Utd	42
16	Wolves	42
17	Tottenham	38
18	Leicester City	25
19	Ipswich Town	22
20	Southampton	12

In [241…
```
import plotly.graph_objects as go
```

In [242…
```
plot_data = League_Table.sort_values(by="Points", ascending=False)
```

In [243…
```
plot_data["Color"] = "Mid-table"
plot_data.loc[plot_data["Team_Pos"] >= 18, "Color"] = "Relegation"
plot_data.loc[plot_data["Team_Pos"] <= 5, "Color"] = "Top 5"
plot_data.loc[plot_data["Team_Pos"] == 1, "Color"] = "Champions"

plt.figure(figsize=(10, 6))
sns.barplot(
    data=plot_data,
    x="Points",
    y="Squad",
    hue="Color",
    dodge=False,
    palette={"Champions": "gold", "Top 5": "green", "Mid-table": "blue",
)

plt.title("Premier League Table", fontsize=14, weight="bold")
plt.xlabel("Points")
plt.ylabel("Team")
plt.legend(title="Outcome")
plt.show()
```



In [244…
```
print(df.columns.tolist())
```

```
['Squad', '# Pl', 'Age', 'MP', 'Gls', 'Ast', 'G+A', 'G-PK', 'PK', 'PKatt',
'xG', 'npxG', 'xAG', 'npxG+xAG', 'PrgC', 'PrgP', 'Gls/90', 'Ast/90', 'G+A/
90', 'G-PK/90', 'G+A-PK', 'xG/90', 'xAG/90', 'xG+xAG', 'npxG/90', 'npxG+xA
G/90', 'GA', 'GA90', 'SoTA', 'Saves', 'Save%', 'W', 'D', 'L', 'CS', 'CS%',
'PKA', 'PKsv', 'PKm', 'Total_Shots', 'SoT', 'SoT%', 'Sh/90', 'SoT/90', 'G/
Sh', 'G/SoT', 'Dist', 'FK_shots', 'npxG/Sh', 'G-xG', 'np:G-xG', 'Cmp', 'Cm
p%', 'Tot_pass_Dist', 'PrgDist_Pass', 's_Cmp', 'Short_p_att', 's_Cmp%', 'm
_Cmp', 'Med_p_att', 'm_Cmp%', 'L-Cmp', 'Long_p_att', 'L_Cmp%', 'xA', 'A-xA
G', 'KP', 'Passes_to_final_3rd', 'PPA', 'CrsPA', 'passes_att', 'Live_ball_
passes', 'Dead', 'FK_passes', 'TB', 'Sw', 'Crs', 'TI', 'CK', 'In', 'Out',
'Str', 'Off', 'SCA', 'SCA90', 'PassLive', 'PassDead', 'Take_ons', 'Sh_to_S
h', 'Fouls_drawn', 'Def_to_shot', 'GCA', 'GCA90', 'PassLive_to_goal', 'Pas
sDead_to_goal', 'Take_on_to_goal', 'Sh_goal_shot', 'Fld_to_goal', 'Def_to_
goal', 'Tkl_total_players', 'Def 3rd', 'Mid 3rd', 'Att 3rd', 'Tkl_dribble
s', 'dribbles_challenged', 'Tkl%', 'Lost', 'Blocks', 'Sh', 'Pass', 'Tkl+In
t', 'Clr', 'Err', 'Poss', 'Touches', 'Def-Pen', 'Def-3rd', 'Mid-3rd', 'Att
-3rd', 'Att-Pen', 'Take_ons_attempted', 'Succ', 'Succ%', 'Tkld', 'Tkld%',
'Carries', 'Tot_carrying_Dist', 'PrgDist_Carrying', 'Carries_to_final_3r
d', 'CPA', 'Mis', 'Dis', 'Rec', 'PrgR', 'CrdY', 'CrdR', '2CrdY', 'Fls', 'F
ld', 'Int', 'TklW', 'PKwon', 'PKcon', 'OG', 'Recov', 'ad_Won', 'ad_Lost',
'ad_Won%', 'Team_Pos', 'Points']
```

In [245…
```
duplicates = df.columns[df.columns.duplicated()]
print(duplicates)
```

```
Index([], dtype='object')
```

In [246…
```
Average_age = round(df['Age'].mean(), 0)
Max_age = round(df['Age'].max(), 0)
Min_age = round(df['Age'].min(), 0)
Median_age = round(df['Age'].median(), 0)
print("Average:", Average_age, "Max:", Max_age, "Min:", Min_age, "Median:
```

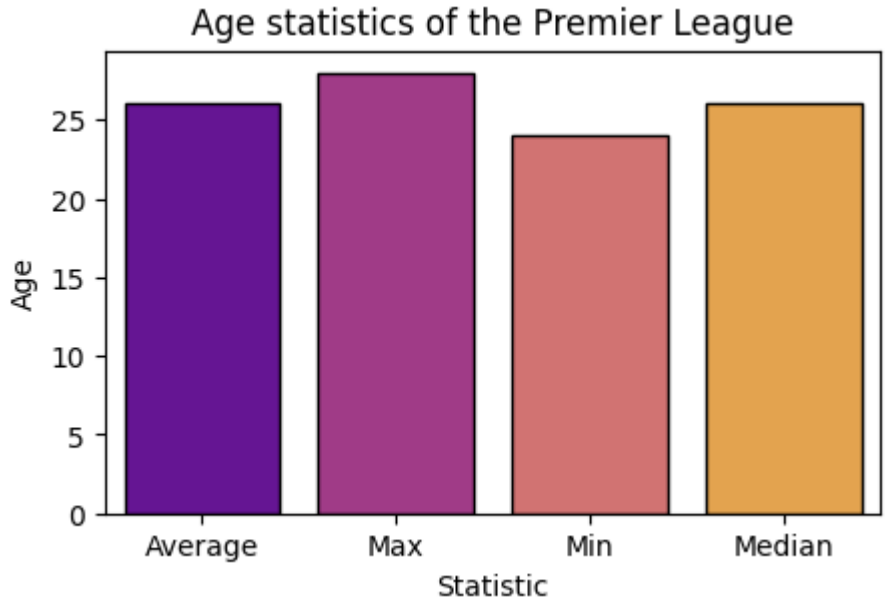
```
Average: 26.0 Max: 28.0 Min: 24.0 Median: 26.0
```

In [247…
```
age_fig = {
    'Average': Average_age,
    'Max': Max_age,
```

```
    'Min': Min_age,
    'Median': Median_age }
```

In [248…
```
Age_df = pd.DataFrame(list(age_fig.items()), columns=['Statistic', 'Age']
```

In [249…
```
plt.figure(figsize=(5,3))
sns.barplot(x="Statistic", y="Age", data=Age_df, hue="Statistic", legend=
    linewidth=1, errorbar=None, edgecolor="0", palette="plasma")
plt.title("Age statistics of the Premier League")
plt.ylabel("Age")
plt.xlabel("Statistic")
plt.show()
```



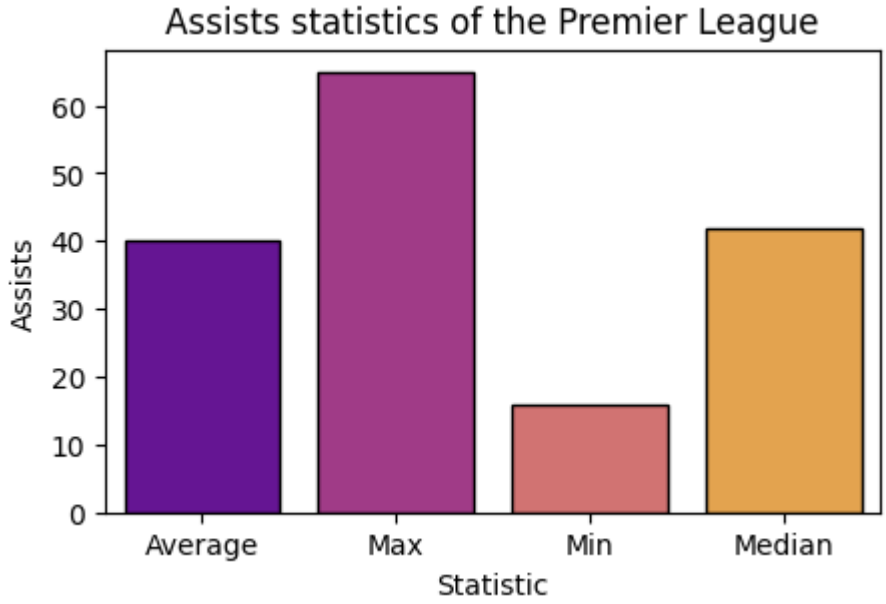
In [250…
```
Average_Ast = round(df['Ast'].mean(), 0)
Max_Ast = round(df['Ast'].max(), 0)
Min_Ast = round(df['Ast'].min(), 0)
Median_Ast = round(df['Ast'].median(), 0)
print("Average:", Average_Ast, "Max:", Max_Ast, "Min:", Min_Ast, "Median:
```

```
Average: 40.0 Max: 65 Min: 16 Median: 42.0
```

In [251…
```
age_fig = {
    'Average': Average_Ast,
    'Max': Max_Ast,
    'Min': Min_Ast,
    'Median': Median_Ast }
```

In [252…
```
Asist_df = pd.DataFrame(list(age_fig.items()), columns=['Statistic', 'Ass
```

In [253…
```
plt.figure(figsize=(5,3))
sns.barplot(x="Statistic", y="Assists", data=Asist_df, hue="Statistic", l
    linewidth=1, errorbar=None, edgecolor="0", palette="plasma")
plt.title("Assists statistics of the Premier League")
plt.ylabel("Assists")
plt.xlabel("Statistic")
plt.show()
```



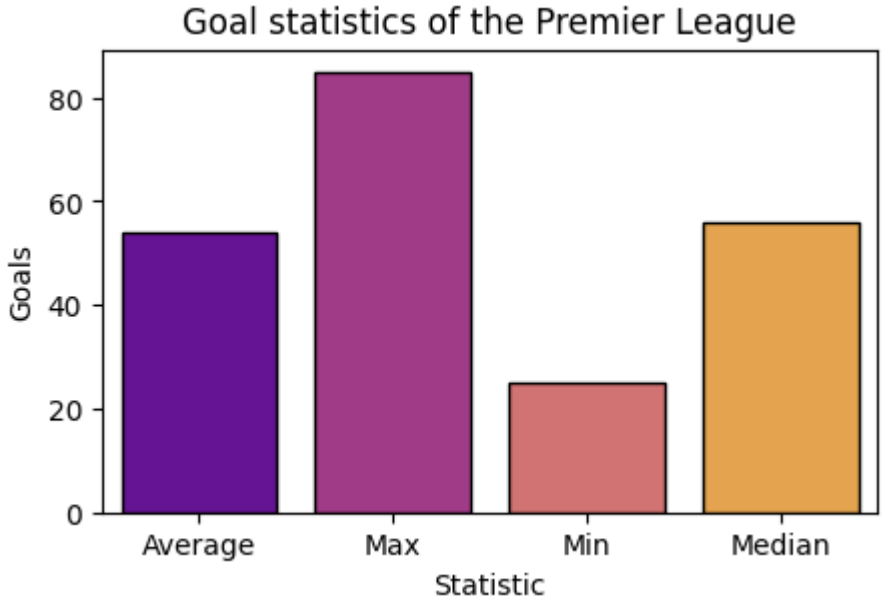
In [254…
```
Average_Gls = round(df['Gls'].mean(), 0)
Max_Gls = round(df['Gls'].max(), 0)
Min_Gls = round(df['Gls'].min(), 0)
Median_Gls = round(df['Gls'].median(), 0)
print("Average:", Average_Gls, "Max:", Max_Gls, "Min:", Min_Gls, "Median:
```

```
Average: 54.0 Max: 85 Min: 25 Median: 56.0
```

In [255…
```
Gls_fig = {
    'Average': Average_Gls,
    'Max': Max_Gls,
    'Min': Min_Gls,
    'Median': Median_Gls }
```

In [256…
```
Gls_df = pd.DataFrame(list(Gls_fig.items()), columns=['Statistic', 'Goals
```

In [257…
```
plt.figure(figsize=(5,3))
sns.barplot(x="Statistic", y="Goals", data=Gls_df, hue="Statistic", legen
    linewidth=1, errorbar=None, edgecolor="0", palette="plasma")
plt.title("Goal statistics of the Premier League")
plt.ylabel("Goals")
plt.xlabel("Statistic")
plt.show()
```



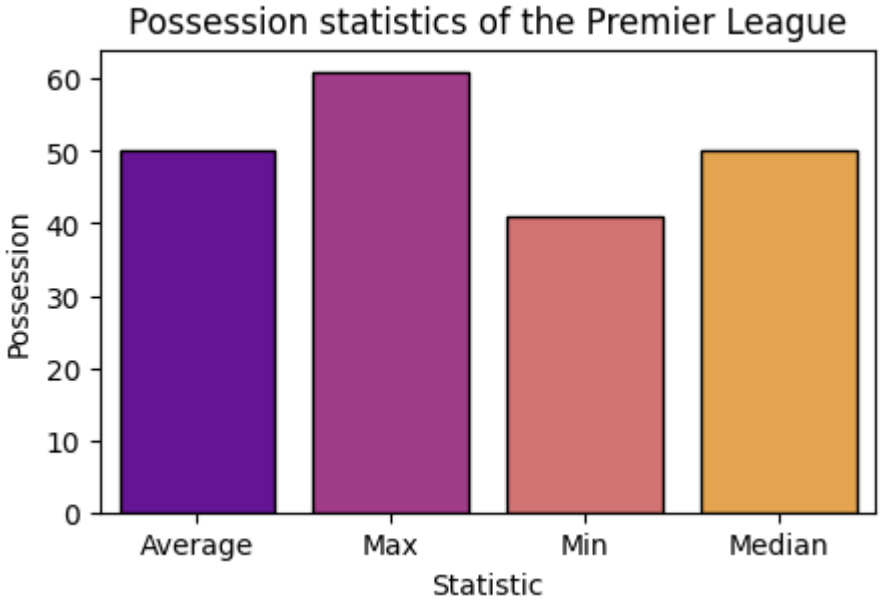
In [258…
```
Average_Poss = round(df['Poss'].mean(), 0)
Max_Poss = round(df['Poss'].max(), 0)
Min_Poss = round(df['Poss'].min(), 0)
Median_Poss = round(df['Poss'].median(), 0)
print("Average:", Average_Poss, "Max:", Max_Poss, "Min:", Min_Poss, "Medi
```

```
Average: 50.0 Max: 61.0 Min: 41.0 Median: 50.0
```

In [259…
```
Poss_fig = {
    'Average': Average_Poss,
    'Max': Max_Poss,
    'Min': Min_Poss,
    'Median': Median_Poss }
```

In [260…
```
Poss_df = pd.DataFrame(list(Poss_fig.items()), columns=['Statistic', 'Pos
```

In [261…
```
plt.figure(figsize=(5,3))
sns.barplot(x="Statistic", y="Possession", data=Poss_df, hue="Statistic",
    linewidth=1, errorbar=None, edgecolor="0", palette="plasma")
plt.title("Possession statistics of the Premier League")
plt.ylabel("Possession")
plt.xlabel("Statistic")
plt.show()
```



Data Analysis

Looking for highest correlated features with the target variable points

In [262…

```
numeric_df = df.select_dtypes(include='number')
correlations = numeric_df.corr()['Points'].sort_values(ascending=False)
top_40 = correlations[1:41]
print(top_40)
```

```
W                     0.988196
G+A/90                0.890123
G+A                   0.889989
Gls                   0.887534
Gls/90                0.887174
G+A-PK                0.883837
GCA                   0.881885
GCA90                 0.881065
G-PK                  0.878990
Ast                   0.877461
G-PK/90               0.877403
Ast/90                0.876509
CS%                   0.849027
CS                    0.848721
xA                    0.846766
npxG/90               0.846514
npxG                  0.845722
xG/90                 0.842607
npxG+xAG/90           0.841235
xG                    0.840910
npxG+xAG              0.840784
xG+xAG                0.840737
SoT                   0.837403
SoT/90                0.836830
xAG/90                0.831799
xAG                   0.830860
Sh/90                 0.825586
Total_Shots           0.825337
SCA                   0.821143
SCA90                 0.821046
PassLive_to_goal      0.820701
Att-Pen               0.813315
PassLive              0.804960
KP                    0.800121
PPA                   0.772174
PrgP                  0.720424
PrgR                  0.718415
CK                    0.703121
Passes_to_final_3rd   0.679415
CPA                   0.674255
Name: Points, dtype: float64
```

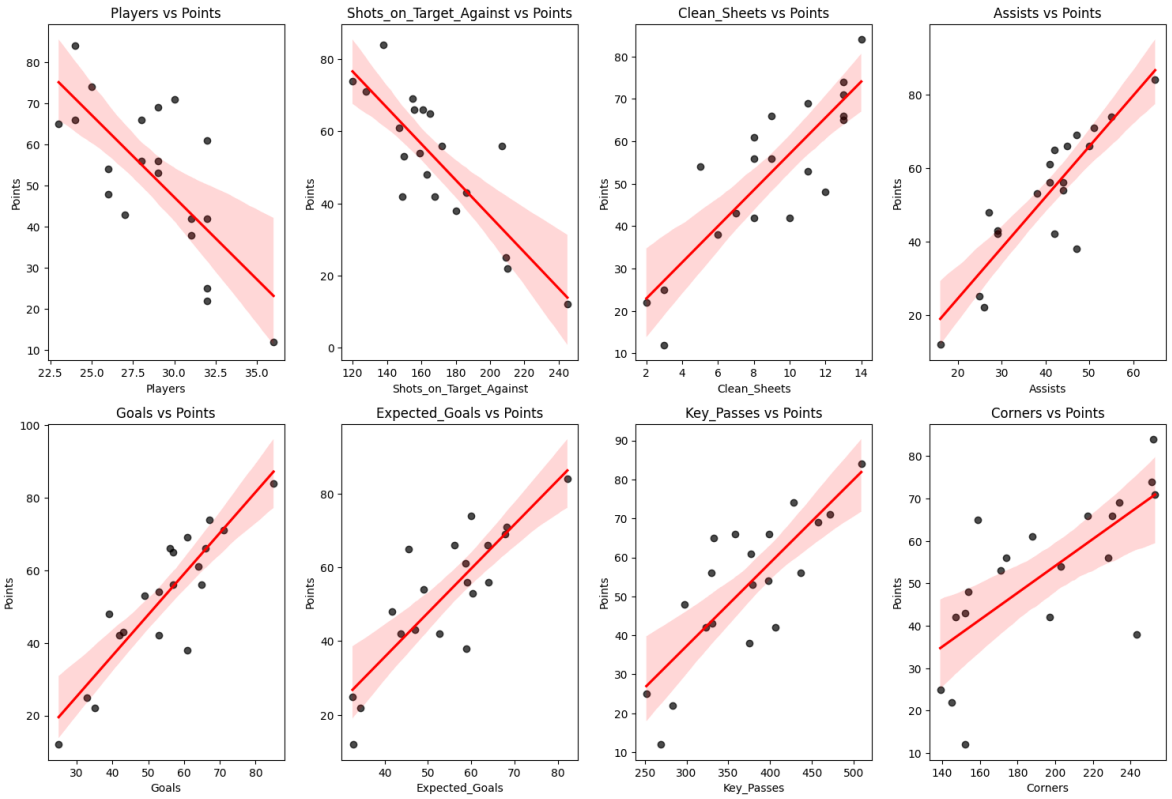
In [263…
```
bottom_10 = correlations.tail(10)
print(bottom_10)
```

```
Def-Pen    -0.650637
# Pl       -0.728120
Sh         -0.748894
SoTA       -0.826534
GA90       -0.908354
GA         -0.908663
L          -0.964464
Team_Pos   -0.970985
MP               NaN
PKm              NaN
Name: Points, dtype: float64
```

In [264…
```
Features = df[['# Pl', 'SoTA', 'CS', 'Ast', 'Gls', 'xG', 'KP', 'CK']]
```

In [265…
```
Features = Features.rename(columns={
    '# Pl': 'Players',
    'SoTA': 'Shots_on_Target_Against',
    'CS': 'Clean_Sheets',
    'Ast': 'Assists',
    'Gls': 'Goals',
    'xG': 'Expected_Goals',
    'KP': 'Key_Passes',
    'CK': 'Corners'
})
```

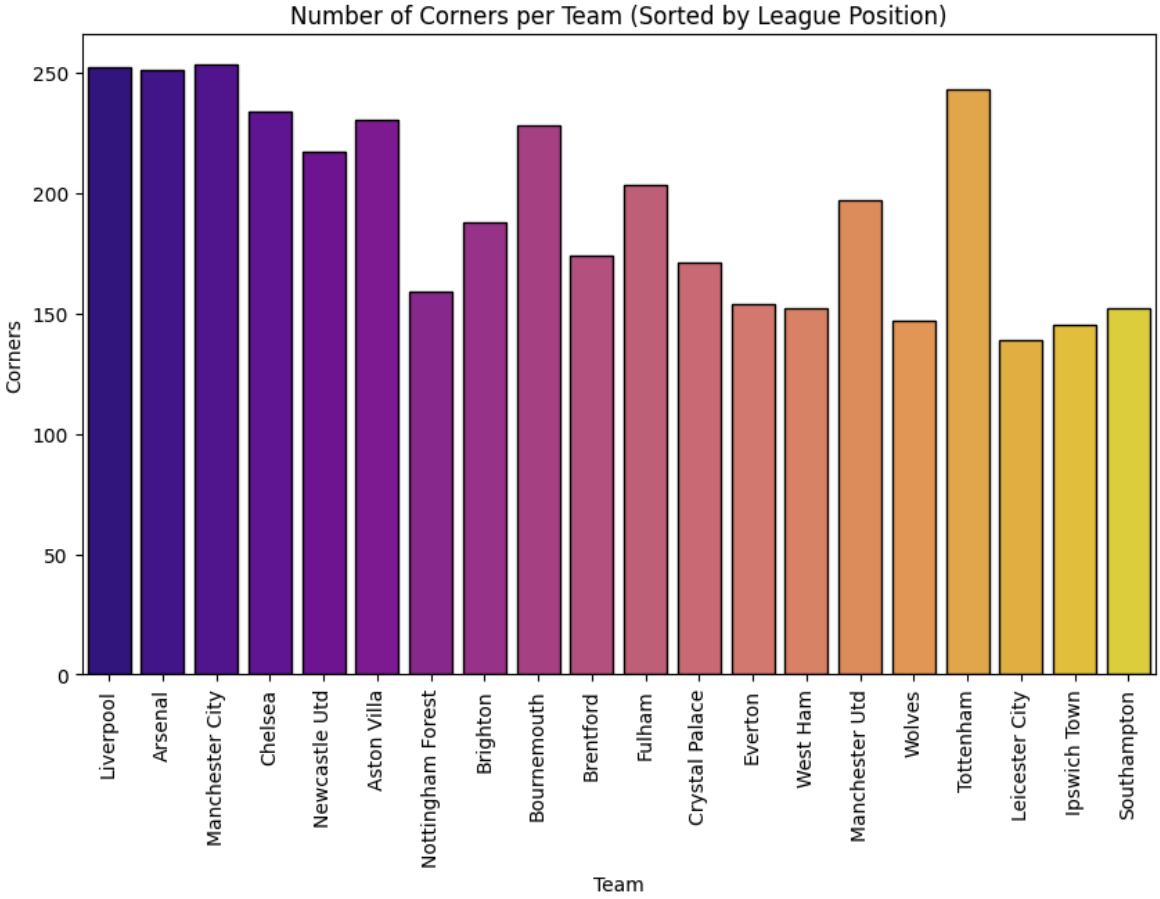
In [266…
```
import math

n_features = len(Features) # All columns used from Features
n_cols = 4 # Number of plots per row (4)
n_rows = math.ceil(n_features / n_cols) # Rounds up rows to fit plots

plt.figure(figsize=(15, 5 * n_rows)) # Height and width of plots - * n_ro

for i, col in enumerate(Features, 1): # Loop through the features to give
    plt.subplot(n_rows, n_cols, i)
    sns.regplot(data=Features.join(df['Points']), x=col, y='Points', scat
    plt.title(f"{col} vs Points")
plt.tight_layout()
plt.show()
```



In [267…
```
Corners_ = df[['CK', 'Squad', 'Team_Pos']]
top_3 = Corners_.sort_values(by='CK', ascending=False).head(3).set_index(
print(top_3)
```

```
             CK           Squad
Team_Pos
3           253  Manchester City
1           252        Liverpool
2           251          Arsenal
```

In [268…
```
plt.figure(figsize=(10,6))

# Sort dataframe by team position
sorted_df = df.sort_values("Team_Pos")

sns.barplot(
    x="Squad", y="CK", data=sorted_df, hue="Squad", legend=False,
    linewidth=1, errorbar=None, edgecolor="0", palette="plasma",
    order=sorted_df["Squad"]   # ensure x-axis follows sorted order
)

plt.title("Number of Corners per Team (Sorted by League Position)")
plt.ylabel("Corners")
plt.xlabel("Team")
plt.xticks(rotation=90)
plt.show()
```



In [269…
```
correlations_CK = numeric_df.corr()['CK'].sort_values(ascending=False)
top_10CK = correlations_CK[1:11]
print(top_10CK)
```

```
PrgP                  0.910106
PrgR                  0.908968
PPA                   0.903125
Att-Pen               0.899957
KP                    0.875022
Passes_to_final_3rd   0.860166
xA                    0.859406
PassLive              0.856911
SCA90                 0.856753
SCA                   0.856741
Name: CK, dtype: float64
```

Defensive Stats

In [270…
```
def_stats = df[['Tkl_total_players', 'Def 3rd', 'Mid 3rd', 'Att 3rd', 'Tk
               'Sh', 'Pass', 'Tkl+Int', 'Clr', 'Err', 'CS']]
```

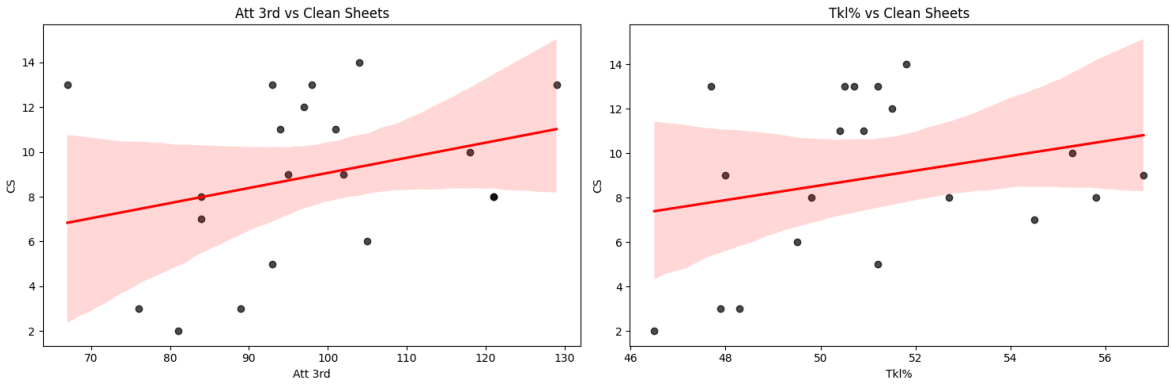
In [271…
```
correlations_CS = def_stats.corr()['CS'].sort_values(ascending=False)
bottom_10CS = correlations_CS[1:]
print(bottom_10CS)
```

```
Att 3rd                0.290051
Tkl%                   0.254070
Tkl_dribbles           0.033264
Mid 3rd               -0.089194
dribbles_challenged   -0.098608
Pass                  -0.100989
Tkl_total_players     -0.127297
Tkl+Int               -0.189693
Def 3rd               -0.193994
Lost                  -0.227874
Clr                   -0.326726
Err                   -0.353336
Blocks                -0.509988
Sh                    -0.660730
Name: CS, dtype: float64
```

The stats that are highest correlated with clean sheets are attacking 3rd tackles and tackle percentage

In [272…
```
Highly_corr = df[['Att 3rd','Tkl%','CS']]
```

In [273…
```
# Exclude 'CS' from the features
features_to_plot = [col for col in Highly_corr.columns if col != 'CS']
n_def = len(features_to_plot)
n_cols = 2
n_rows = math.ceil(n_def / n_cols)

plt.figure(figsize=(15, 5 * n_rows))
for i, col in enumerate(features_to_plot, 1):
    plt.subplot(n_rows, n_cols, i)
    sns.regplot(data=Highly_corr, x=col, y='CS', scatter_kws={'alpha':0.7
    plt.title(f"{col} vs Clean Sheets")
plt.tight_layout()
plt.show()
```



In [274…
```
Press_ = df[['Att 3rd', 'Squad', 'Team_Pos']]
top_3_press = Press_.sort_values(by='Att 3rd', ascending=False).head(3).s
```

```
print(top_3_press)
```

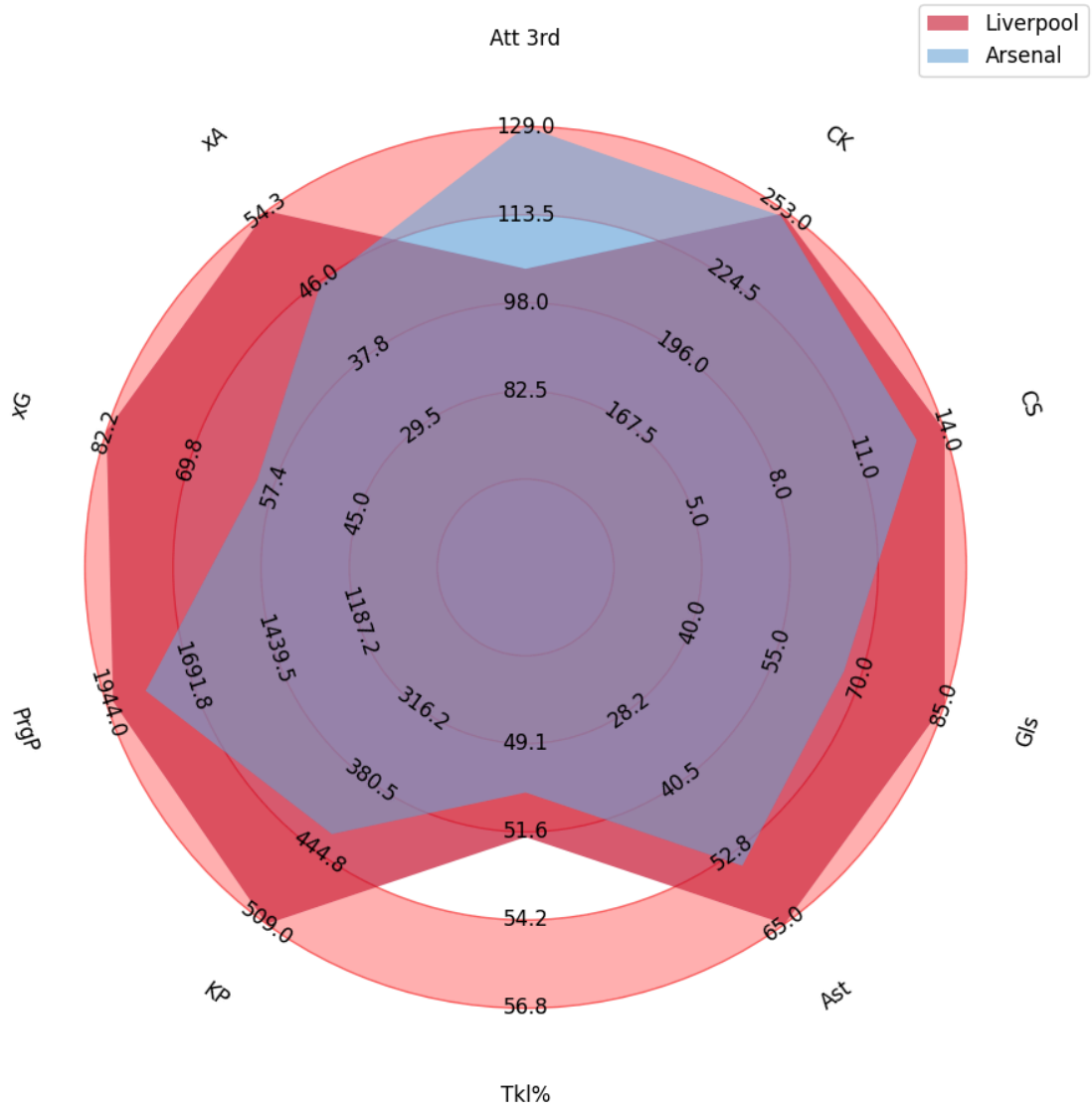
```
          Att 3rd      Squad
Team_Pos
2             129    Arsenal
10            121  Brentford
8             121   Brighton
```

In [275…
```
Liv_Press = df.loc[df['Squad'] == 'Liverpool', ['Squad', 'Att 3rd']].set_
print(Liv_Press)
```

```
           Att 3rd
Squad
Liverpool      104
```

Data Visualisation

In [276…
```
params = ['Att 3rd', 'CK', 'CS', 'Gls', 'Ast', 'Tkl%', 'KP', 'PrgP', 'xG'

low = df[params].min().tolist()
high = df[params].max().tolist()

Liv = "Liverpool"
Ars = "Arsenal"
team_values = df.loc[df['Squad'] == Liv, params].iloc[0].tolist()
team2_values = df.loc[df['Squad'] == Ars, params].iloc[0].tolist()

radar = Radar(params, low, high,
              round_int=[False]*len(params),
              num_rings=4,
              ring_width=1, center_circle_radius=1)

# Setup Fig
fig, ax = radar.setup_axis()
rings_inner = radar.draw_circles(ax=ax, facecolor='#ffb2b2', edgecolor='#

# Liverpool
radar_output1 = radar.draw_radar(team_values, ax=ax,
                                 kwargs_radar={'facecolor': '#c8102e', 'a
                                 kwargs_rings={'facecolor': '#c8102e', 'a

# Arsenal
radar_output2 = radar.draw_radar(team2_values, ax=ax,
                                 kwargs_radar={'facecolor': '#6CABDD', 'a
                                 kwargs_rings={'facecolor': '#6CABDD', 'a



range_labels = radar.draw_range_labels(ax=ax, fontsize=12)
param_labels = radar.draw_param_labels(ax=ax, fontsize=12)

ax.legend([Liv, Ars], loc='upper right', fontsize=12);
```



Comparison of different stats between Arsenal and Liverpool

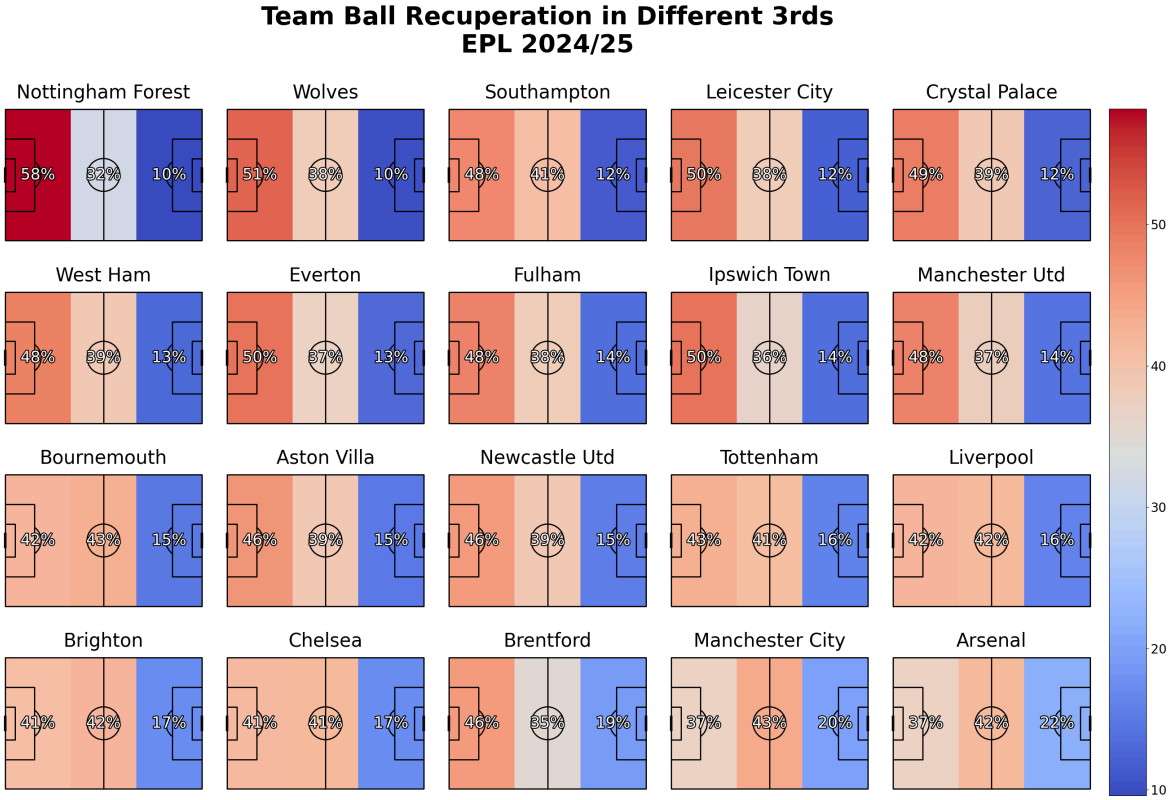
In [277…
```
from urllib.request import urlopen

import matplotlib.patheffects as path_effects
import matplotlib.pyplot as plt
import pandas as pd
from PIL import Image

from mplsoccer import Pitch, FontManager, add_image
```

In [278…
```
touches_cols = ['Def 3rd', 'Mid 3rd', 'Att 3rd']
df_total = pd.DataFrame(df[touches_cols].sum())
df_total.columns = ['total']
df_total = df_total.T
df_total = df_total.divide(df_total.sum(axis=1), axis=0) * 100
```

In [279…
```
path_eff = [path_effects.Stroke(linewidth=3, foreground='black'),
            path_effects.Normal()]
```

In [280…
```
touches_cols = ['Def 3rd', 'Mid 3rd', 'Att 3rd']
path_eff = [path_effects.Stroke(linewidth=3, foreground='black'),
            path_effects.Normal()]
```

```
df[touches_cols] = df[touches_cols].divide(df[touches_cols].sum(axis=1),

# Optional: sort teams by attacking/defensive touches
df.sort_values(['Att 3rd', 'Def 3rd'], ascending=[True, False], inplace=T

# Setup pitch and grid
pitch = Pitch(line_zorder=2, line_color='black', pad_top=20)
GRID_HEIGHT = 0.8
CBAR_WIDTH = 0.03

fig, axs = pitch.grid(nrows=4, ncols=5, figheight=20,
                      grid_width=0.88, left=0.025,
                      endnote_height=0.03, endnote_space=0,
                      axis=False,
                      title_space=0.02, title_height=0.06, grid_height=GR
fig.set_facecolor('white')

# Prepare bin statistic (3x1 grid per team)
bin_statistic = pitch.bin_statistic([0], [0], statistic='count', bins=(3,

# Colour normalisation
vmin = df[touches_cols].min().min()
vmax = df[touches_cols].max().max()

# Loop over teams and axes
teams = df['Squad'].values
for i, ax in enumerate(axs['pitch'].flat[:len(teams)]):
    # Plot team name above pitch
    ax.text(60, -10, teams[i], ha='center', va='center', fontsize=30)

    # Extract team values and reshape to bin_statistic shape
    team_values = df.loc[df['Squad'] == teams[i], touches_cols].values.fl
    bin_statistic['statistic'] = team_values.reshape(bin_statistic['stati

    # Draw heatmap and labels
    heatmap = pitch.heatmap(bin_statistic, ax=ax, cmap='coolwarm', vmin=v
    pitch.label_heatmap(bin_statistic, color='white', path_effects=path_e
                        fontsize=25, ax=ax, str_format='{0:.0f}%', ha='ce

# Colorbar
cbar_bottom = axs['pitch'][-1, 0].get_position().y0
cbar_left = axs['pitch'][0, -1].get_position().x1 + 0.01
ax_cbar = fig.add_axes((cbar_left, cbar_bottom, CBAR_WIDTH, GRID_HEIGHT -
cbar = plt.colorbar(heatmap, cax=ax_cbar)
for label in cbar.ax.get_yticklabels():
    label.set_fontsize(20)

# Title
axs['title'].text(0.5, 0.5, 'Team Ball Recuperation in Different 3rds\nEP
                  ha='center', va='center', fontsize=40, weight='bold')

plt.show()
```



In [281…
```
df[touches_cols] = df[touches_cols].values - df_total.values
```

In [282…
```
pitch = Pitch(line_zorder=2, line_color='black', pad_top=20)
GRID_HEIGHT = 0.8
CBAR_WIDTH = 0.03

fig, axs = pitch.grid(nrows=4, ncols=5, figheight=20,
                      grid_width=0.88, left=0.025,
                      endnote_height=0.03, endnote_space=0,
                      axis=False,
                      title_space=0.02, title_height=0.06, grid_height=GR
fig.set_facecolor('white')

# Prepare bin statistic (3x1 grid per team)
bin_statistic = pitch.bin_statistic([0], [0], statistic='count', bins=(3,

# Color normalization
vmin = df[touches_cols].min().min()
vmax = df[touches_cols].max().max()

# Loop over teams and axes
teams = df['Squad'].values
for i, ax in enumerate(axs['pitch'].flat[:len(teams)]):
    # Plot team name above pitch
    ax.text(60, -10, teams[i], ha='center', va='center', fontsize=30)

    # Extract team values and reshape to bin_statistic shape
    team_values = df.loc[df['Squad'] == teams[i], touches_cols].values.fl
    bin_statistic['statistic'] = team_values.reshape(bin_statistic['stati

    # Draw heatmap and labels
    heatmap = pitch.heatmap(bin_statistic, ax=ax, cmap='coolwarm', vmin=v
    pitch.label_heatmap(bin_statistic, color='white', path_effects=path_e
                        fontsize=25, ax=ax, str_format='{0:.0f}%', ha='ce
```

```
# Colorbar
cbar_bottom = axs['pitch'][-1, 0].get_position().y0
cbar_left = axs['pitch'][0, -1].get_position().x1 + 0.01
ax_cbar = fig.add_axes((cbar_left, cbar_bottom, CBAR_WIDTH, GRID_HEIGHT -
cbar = plt.colorbar(heatmap, cax=ax_cbar)
for label in cbar.ax.get_yticklabels():
    label.set_fontsize(20)

# Title
axs['title'].text(0.5, 0.5, 'Ball Recuperation Percentage in Comparison t
                  ha='center', va='center', fontsize=40, weight='bold')

plt.show()
```