premier league analysis

December 15, 2024

[1]:
```
import pandas as pd
import numpy as np
from sqlalchemy import create_engine, text
import matplotlib.pyplot as plt
import seaborn as sns
import math
```

[2]:
```
data=pd.read_csv("D:/epl1.csv")
data.head(5)
```

[2]:
```
  league  year h_a        xG       xGA      npxG     npxGA  deep  \
0    EPL  2020   h  0.805270  0.849709  0.805270  0.088540    17
1    EPL  2020   a  2.032220  0.534675  2.032220  0.534675    10
2    EPL  2020   h  3.076260  1.657050  3.076260  1.657050     7
3    EPL  2020   a  0.873776  0.671595  0.873776  0.671595     7
4    EPL  2020   h  1.501250  2.376950  1.501250  2.376950     7

   deep_allowed  scored  …  ppda_coef  ppda_att ppda_def oppda_coef  \
0             2       1  …   4.450000        89       20  17.642857
1             5       3  …   9.303030       307       33   5.958333
2            18       7  …  14.600000       365       25   4.760000
3             4       1  …   9.217391       212       23   8.750000
4            20       0  …  13.235294       225       17   3.647059

   oppda_att  oppda_def         team    xG_diff  xGA_diff  xpts_diff
0        247         14  Aston Villa -0.194730  0.849709    -1.8399
1        143         24  Aston Villa -0.967780  0.534675    -0.5369
2        119         25  Aston Villa -3.923740 -0.342950    -0.7431
3        210         24  Aston Villa -0.126224  0.671595    -1.4709
4        124         34  Aston Villa  1.501250 -0.623050     0.8236

[5 rows x 29 columns]
```

[3]:
```
# MySQL Database connection parameters (no password)
username = 'root'  # Replace with your MySQL username (e.g., 'root')
password = '[redacted]'                # Leave it as an empty string if no␣
 ↪password is set
host = 'localhost'          # Replace with your MySQL host (localhost is common)
```

```
port = '3306'                # Default MySQL port
database = 'dst_project'     # Desired database name

# Create a connection string to MySQL using SQLAlchemy (without specifying the␣
↪database)
connection_string = f'mysql+mysqlconnector://{username}:{password}@{host}:
↪{port}'
engine = create_engine(connection_string)

# Create the database 'DST_project' if it doesn't exist
with engine.connect() as conn:
    conn.execute(text(f"CREATE DATABASE IF NOT EXISTS {database}"))
    conn.execute(text(f"USE {database}"))



# Create a new connection (this time specifying the database)
connection_string_with_db = f'mysql+mysqlconnector://{username}:
↪{password}@{host}:{port}/{database}'
engine_with_db = create_engine(connection_string_with_db)

# Write the DataFrame to the MySQL database, create a table named 'Premier␣
↪League'
data.to_sql('premier_league', con=engine_with_db, if_exists='replace',␣
↪index=False)
```

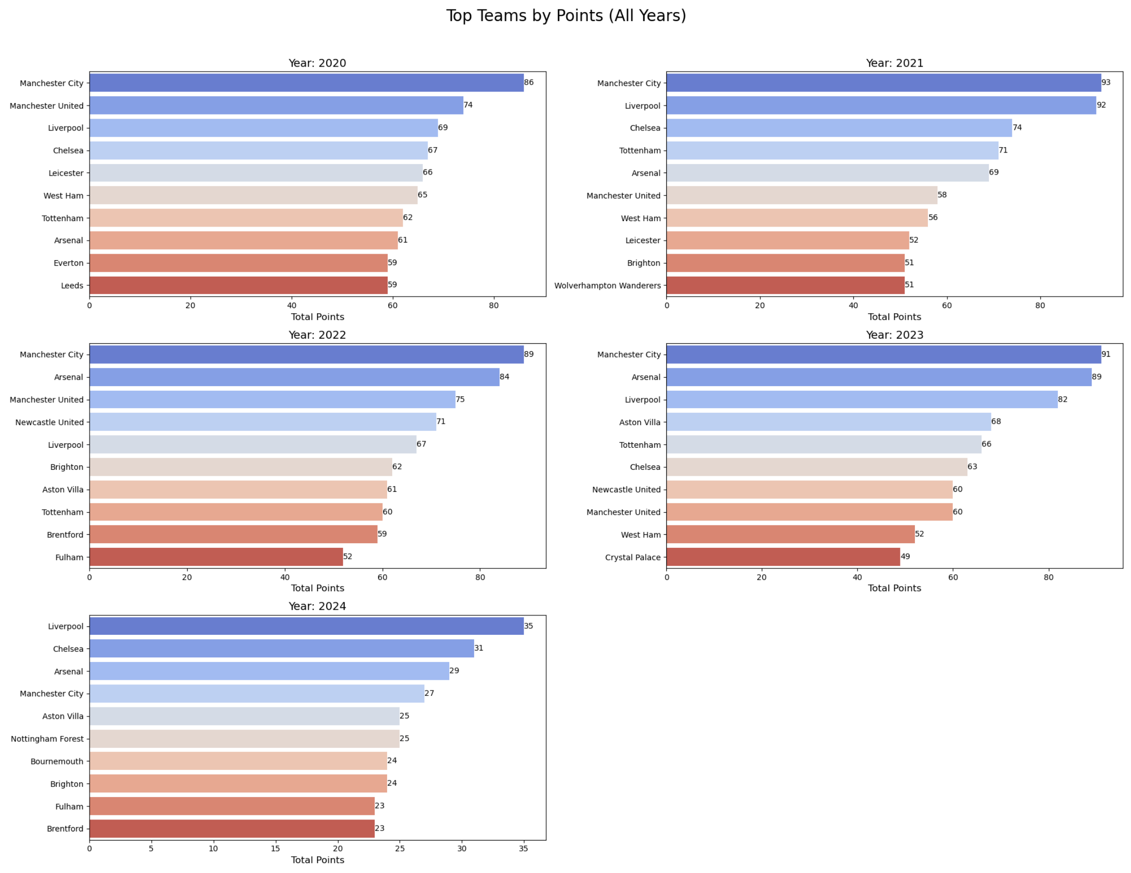
[3]:
```
3338
```

[4]:
```
query = "SELECT * FROM premier_league"
df = pd.read_sql(query, con=engine)
print(df.head(5))
```

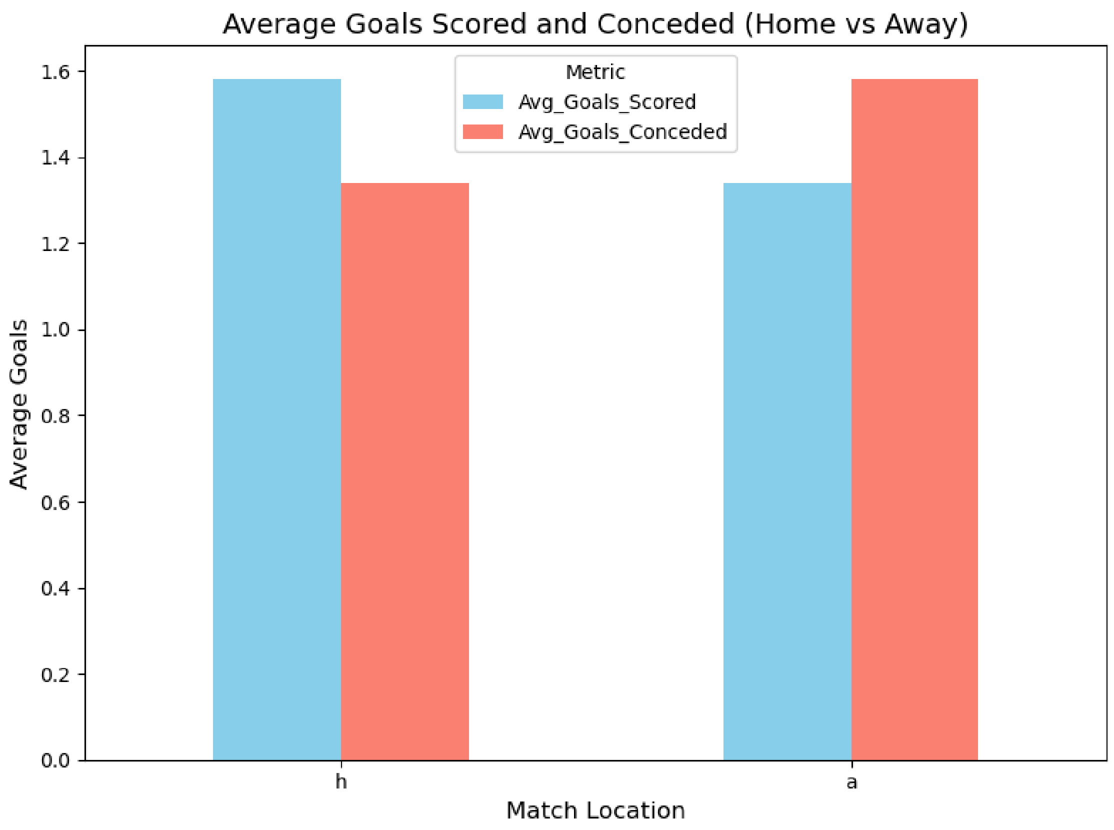
```
  league  year h_a        xG       xGA      npxG     npxGA  deep  \
0    EPL  2020   h  0.805270  0.849709  0.805270  0.088540    17
1    EPL  2020   a  2.032220  0.534675  2.032220  0.534675    10
2    EPL  2020   h  3.076260  1.657050  3.076260  1.657050     7
3    EPL  2020   a  0.873776  0.671595  0.873776  0.671595     7
4    EPL  2020   h  1.501250  2.376950  1.501250  2.376950     7

   deep_allowed  scored  …  ppda_coef  ppda_att ppda_def oppda_coef  \
0             2       1  …   4.450000        89       20  17.642857
1             5       3  …   9.303030       307       33   5.958333
2            18       7  …  14.600000       365       25   4.760000
3             4       1  …   9.217391       212       23   8.750000
4            20       0  …  13.235294       225       17   3.647059

   oppda_att  oppda_def         team   xG_diff  xGA_diff  xpts_diff
0        247         14  Aston Villa -0.194730  0.849709    -1.8399
```

```
1        143         24  Aston Villa -0.967780  0.534675    -0.5369
2        119         25  Aston Villa -3.923740 -0.342950    -0.7431
3        210         24  Aston Villa -0.126224  0.671595    -1.4709
4        124         34  Aston Villa  1.501250 -0.623050     0.8236

[5 rows x 29 columns]
```

[5]:
```
query_top_teams = """
SELECT year, team, SUM(pts) AS total_points
FROM premier_league
GROUP BY year, team
ORDER BY year, total_points DESC;
"""
df_top_teams = pd.read_sql(query_top_teams, con=engine_with_db)

# Group data by year and sort for top teams
grouped = df_top_teams.groupby('year')
years = sorted(df_top_teams['year'].unique())

# Number of rows and columns for the grid layout
cols = 2  # Number of columns in the grid
rows = 3  # Number of rows in the grid

# Create the figure
fig, axes = plt.subplots(rows, cols, figsize=(20, rows * 5))
fig.suptitle("Top Teams by Points (All Years)", fontsize=20, y=1.02)

# Flatten axes for easy iteration
axes = axes.flatten()

# Set a color palette
sns.set_palette("Blues_r")

# Plot each year's data
for i, (year, group) in enumerate(grouped):
    ax = axes[i]
    top_teams = group.sort_values('total_points', ascending=False).head(10)

    sns.barplot(
        x='total_points',  # 'total_points' on the x-axis (horizontal bars)
        y='team',           # 'team' on the y-axis (vertical labels)
        data=top_teams,
        ax=ax,
        palette='coolwarm'
    )

    # Customize the subplot
    ax.set_title(f"Year: {year}", fontsize=14)
    ax.set_xlabel("Total Points", fontsize=12)
    ax.set_ylabel("")

    # Add data labels
    for index, value in enumerate(top_teams['total_points']):
        ax.text(value, index, f'{value:.0f}', va='center')

# Turn off unused subplots if there are fewer years than the grid cells
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

# Adjust layout to give space for the last row plot
plt.subplots_adjust(hspace=0.4, wspace=0.3)

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

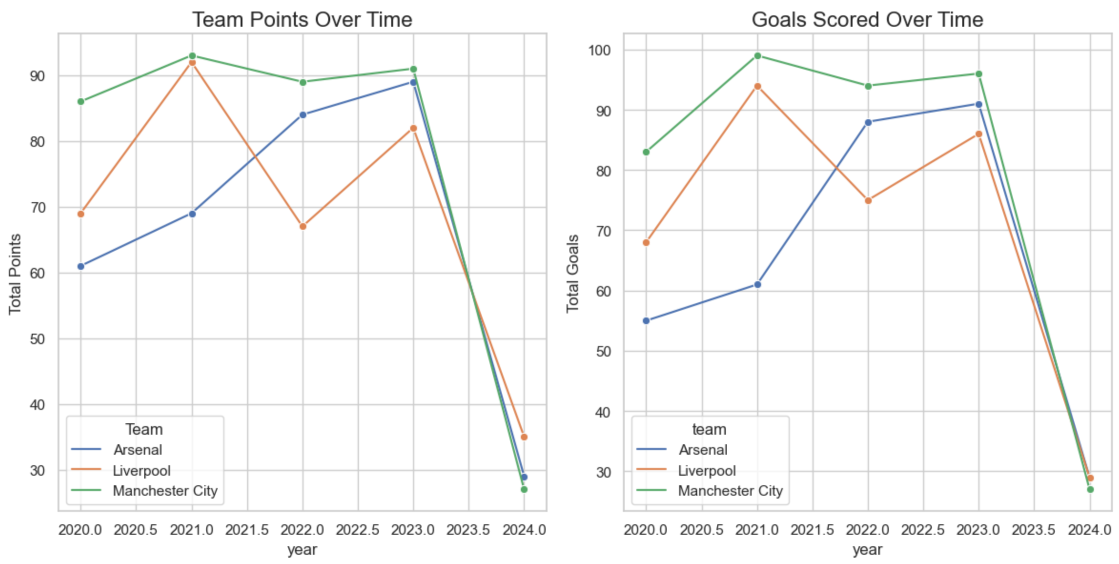


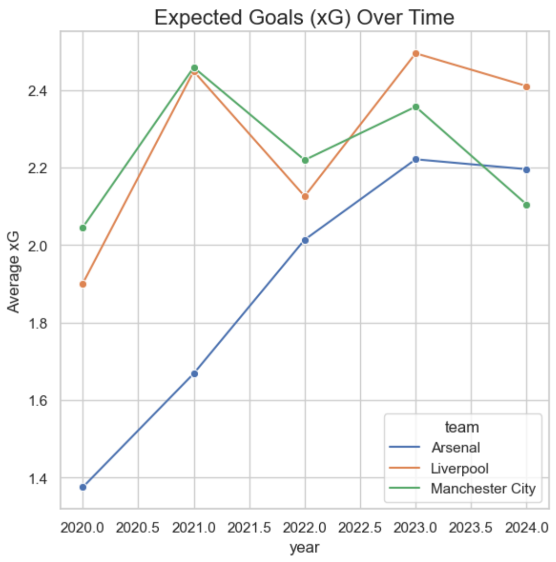
```
[6]: query_home_away = """
SELECT
    h_a AS Match_Location,
    AVG(scored) AS Avg_Goals_Scored,
    AVG(missed) AS Avg_Goals_Conceded
FROM
    premier_league
GROUP BY
    h_a
"""

# Execute the query and fetch results
df_home_away = pd.read_sql(query_home_away, con=engine_with_db)


# Visualization for Home vs Away Performance
fig, ax = plt.subplots(figsize=(8, 6))

# Bar plot for average goals scored and conceded
df_home_away.set_index('Match_Location')[['Avg_Goals_Scored',␣
 ↪'Avg_Goals_Conceded']].plot(kind='bar', ax=ax, color=['skyblue', 'salmon'])

# Customize the plot
ax.set_title('Average Goals Scored and Conceded (Home vs Away)', fontsize=14)
ax.set_ylabel('Average Goals', fontsize=12)
ax.set_xlabel('Match Location', fontsize=12)
ax.legend(title='Metric', fontsize=10)
plt.xticks(rotation=0)
plt.tight_layout()

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



[7]:
```
# SQL query to fetch team performance metrics over time
query_team_trends = """
SELECT year, team, SUM(pts) AS total_points, SUM(scored) AS total_scored,␣
  ↪AVG(xG) AS avg_xG
FROM premier_league
GROUP BY year, team
ORDER BY year, team;
"""

# Fetch the data
df_team_trends = pd.read_sql(query_team_trends, con=engine_with_db)

# Filter for specific teams (optional)
selected_teams = ['Manchester City', 'Liverpool', 'Arsenal']  # Example teams
df_filtered = df_team_trends[df_team_trends['team'].isin(selected_teams)]

# Plotting trends in a grid layout (2x2)
sns.set(style="whitegrid")
fig, axes = plt.subplots(2, 2, figsize=(12, 12))
```

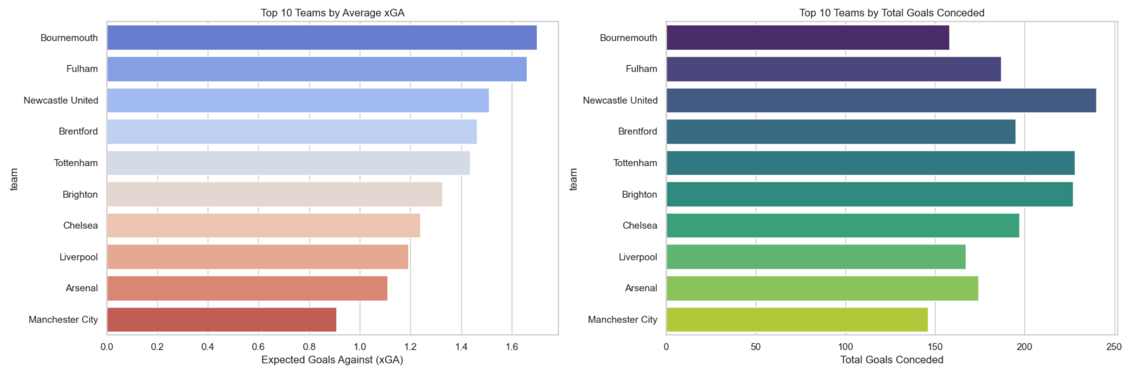
```
# Points Trend
sns.lineplot(
    data=df_filtered, x='year', y='total_points', hue='team', ax=axes[0, 0],␣
 ↪marker='o'
)
axes[0, 0].set_title('Team Points Over Time', fontsize=16)
axes[0, 0].set_ylabel('Total Points')
axes[0, 0].legend(title='Team')

# Goals Scored Trend
sns.lineplot(
    data=df_filtered, x='year', y='total_scored', hue='team', ax=axes[0, 1],␣
 ↪marker='o'
)
axes[0, 1].set_title('Goals Scored Over Time', fontsize=16)
axes[0, 1].set_ylabel('Total Goals')

# Expected Goals (xG) Trend
sns.lineplot(
    data=df_filtered, x='year', y='avg_xG', hue='team', ax=axes[1, 0],␣
 ↪marker='o'
)
axes[1, 0].set_title('Expected Goals (xG) Over Time', fontsize=16)
axes[1, 0].set_ylabel('Average xG')

# An empty subplot (you can use it for additional visualizations)
axes[1, 1].axis('off')  # Turns off the unused subplot
axes[1, 1].set_title('')

# Adjust layout to prevent overlapping
plt.tight_layout()
plt.show()
```





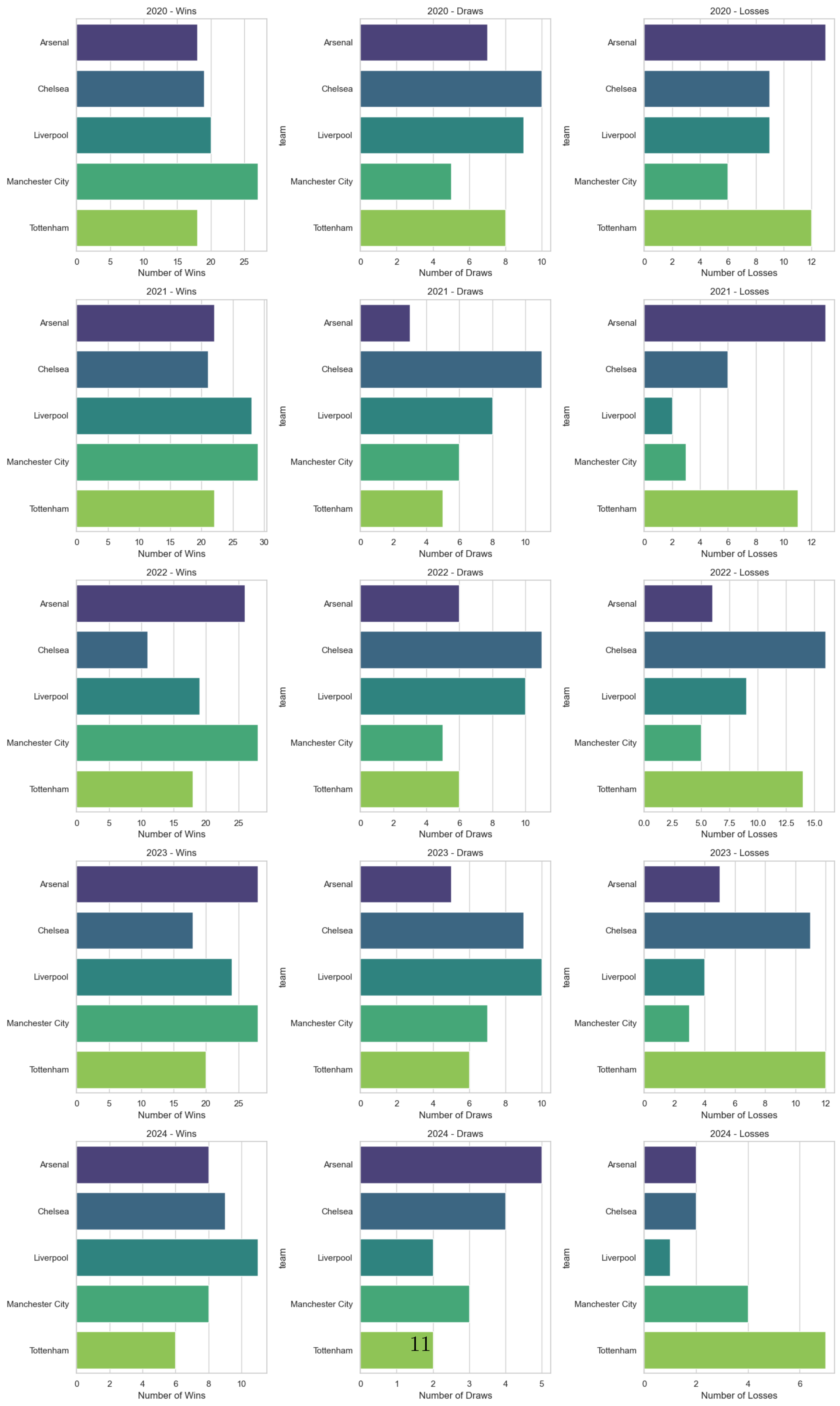
```
[8]: # SQL query to calculate average xGA and total goals conceded
query_top_teams_analysis = """
SELECT team, AVG(xGA) AS avg_xGA, SUM(missed) AS total_goals_conceded
FROM premier_league
WHERE team IN ('Chelsea', 'Liverpool', 'Arsenal', 'Bournemouth', 'Fulham',␣
  ↪'Brentford', 'Newcastle United', 'Brighton', 'Tottenham', 'Manchester City')
GROUP BY team
ORDER BY avg_xGA DESC, total_goals_conceded DESC;
"""

# Fetch the data
```

```
df_top_teams_analysis = pd.read_sql(query_top_teams_analysis,␣
  ↪con=engine_with_db)

# Visualize the data
sns.set(style="whitegrid")
fig, axes = plt.subplots(1, 2, figsize=(18, 6))

# Plot Average xGA
sns.barplot(data=df_top_teams_analysis, x='avg_xGA', y='team', ax=axes[0],␣
  ↪palette='coolwarm')
axes[0].set_title('Top 10 Teams by Average xGA')
axes[0].set_xlabel('Expected Goals Against (xGA)')

# Plot Total Goals Conceded
sns.barplot(data=df_top_teams_analysis, x='total_goals_conceded', y='team',␣
  ↪ax=axes[1], palette='viridis')
axes[1].set_title('Top 10 Teams by Total Goals Conceded')
axes[1].set_xlabel('Total Goals Conceded')

plt.tight_layout()
plt.show()
```



[9]:
```
# SQL query to calculate wins, draws, and losses for top 10 teams
query_top_teams_analysis = """
SELECT
    team,
    year,
    COUNT(CASE WHEN result = 'w' THEN 1 END) AS wins,
    COUNT(CASE WHEN result = 'd' THEN 1 END) AS draws,
    COUNT(CASE WHEN result = 'l' THEN 1 END) AS losses
FROM
    premier_league
WHERE
```

```
    team IN ('Manchester City', 'Liverpool', 'Arsenal', 'Chelsea', 'Tottenham')
GROUP BY
    year, team
ORDER BY
    year, team;
"""

# Fetch the data
df_top_teams_analysis = pd.read_sql(query_top_teams_analysis,␣
 ↪con=engine_with_db)


# Set the Seaborn style
sns.set(style="whitegrid")

years = [2020, 2021, 2022, 2023, 2024]

fig, axes = plt.subplots(5, 3, figsize=(15, 25))

# Loop through each year to create subplots individually
for idx, year in enumerate(years):
    df_year = df_top_teams_analysis[df_top_teams_analysis['year'] == year]

    # Plot Wins
    sns.barplot(data=df_year, x='wins', y='team', ax=axes[idx, 0],␣
 ↪palette='viridis')
    axes[idx, 0].set_title(f'{year} - Wins')
    axes[idx, 0].set_xlabel('Number of Wins')
    axes[idx, 0].set_ylabel('')

    # Plot Draws
    sns.barplot(data=df_year, x='draws', y='team', ax=axes[idx, 1],␣
 ↪palette='viridis')
    axes[idx, 1].set_title(f'{year} - Draws')
    axes[idx, 1].set_xlabel('Number of Draws')

    # Plot Losses
    sns.barplot(data=df_year, x='losses', y='team', ax=axes[idx, 2],␣
 ↪palette='viridis')
    axes[idx, 2].set_title(f'{year} - Losses')
    axes[idx, 2].set_xlabel('Number of Losses')

plt.tight_layout()
plt.show()
```


2020 - Wins
2020 - Draws
2020 - Losses
2021 - Wins
2021 - Draws
2021 - Losses
2022 - Wins
2022 - Draws
2022 - Losses
2023 - Wins
2023 - Draws
2023 - Losses
2024 - Wins
2024 - Draws
2024 - Losses
Arsenal
Chelsea
Liverpool
Manchester City
Tottenham
team
Number of Wins
Number of Draws
Number of Losses

```
[ ]:
```