# CS418

## EDA - Past Match-up analysis

In [18]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import get_data
import time
import requests
from bs4 import BeautifulSoup
from nba_api.stats.endpoints import leaguegamefinder
```

In [22]:
```
# Fetch the regular season schedule for the 2024-25 season
df_schedule = get_data.fetch_regular_season_schedule(season='2024-25')
df_schedule = df_schedule[df_schedule['WL'].isin(['W', 'L'])]
# Display the first few rows of the DataFrame
df_schedule.head()
```

Out[22]:

| | SEASON_ID | TEAM_ID | TEAM_ABBREVIATION | TEAM_NAME | GAME_ID | GAME_DATE | MATC |
|---|---|---|---|---|---|---|---|
| **2** | 22024 | 1610612763 | MEM | Memphis Grizzlies | 0022401093 | 2025-03-31 | MEI |
| **3** | 22024 | 1610612760 | OKC | Oklahoma City Thunder | 0022401094 | 2025-03-31 | OK |
| **4** | 22024 | 1610612751 | BKN | Brooklyn Nets | 0022401095 | 2025-03-31 | BI |
| **5** | 22024 | 1610612741 | CHI | Chicago Bulls | 0022401094 | 2025-03-31 | C |
| **6** | 22024 | 1610612753 | ORL | Orlando Magic | 0022401091 | 2025-03-31 | OF |

5 rows × 28 columns

In [23]:
```
df_schedule.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2250 entries, 2 to 2251
Data columns (total 28 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   SEASON_ID          2250 non-null   object
 1   TEAM_ID            2250 non-null   int64
 2   TEAM_ABBREVIATION  2250 non-null   object
 3   TEAM_NAME          2250 non-null   object
 4   GAME_ID            2250 non-null   object
 5   GAME_DATE          2250 non-null   datetime64[ns]
 6   MATCHUP            2250 non-null   object
 7   WL                 2250 non-null   object
 8   MIN                2250 non-null   int64
 9   PTS                2250 non-null   int64
 10  FGM                2250 non-null   int64
 11  FGA                2250 non-null   int64
 12  FG_PCT             2250 non-null   float64
 13  FG3M               2250 non-null   int64
 14  FG3A               2250 non-null   int64
 15  FG3_PCT            2250 non-null   float64
 16  FTM                2250 non-null   int64
 17  FTA                2250 non-null   int64
 18  FT_PCT             2250 non-null   float64
 19  OREB               2250 non-null   int64
 20  DREB               2250 non-null   int64
 21  REB                2250 non-null   int64
 22  AST                2250 non-null   int64
 23  STL                2250 non-null   int64
 24  BLK                2250 non-null   int64
 25  TOV                2250 non-null   int64
 26  PF                 2250 non-null   int64
 27  PLUS_MINUS         2250 non-null   float64
dtypes: datetime64[ns](1), float64(4), int64(17), object(6)
memory usage: 509.8+ KB
```

## DATA DESCRIPTION

SEASON_ID: The ID representing the NBA season (e.g., "22024" for the 2024-25 season).

TEAM_ID: A unique identifier for each NBA team.

TEAM_ABBREVIATION: The abbreviated name of the team (e.g., "LAL" for Los Angeles Lakers).

TEAM_NAME: The full name of the team (e.g., "Los Angeles Lakers").

GAME_ID: A unique identifier for each NBA game played.

GAME_DATE: The date on which the game was played (datetime format).

MATCHUP: Description of the game matchup, including home vs. away status (e.g., "LAL vs. BOS" or "LAL @ BOS").

WL: Result of the game for the team - "W" for win and "L" for loss.

MIN: Total minutes played by the team during the game.

PTS: Total points scored by the team in the game.

FGM: Field Goals Made - Total successful field goals made by the team.

FGA: Field Goals Attempted - Total field goals attempted by the team.

FG_PCT: Field Goal Percentage - The ratio of FGM to FGA expressed as a percentage.

FG3M: Three-Point Field Goals Made - Total successful three-point shots made by the team.

FG3A: Three-Point Field Goals Attempted - Total three-point shots attempted by the team.

FG3_PCT: Three-Point Field Goal Percentage - The ratio of FG3M to FG3A expressed as a percentage.

FTM: Free Throws Made - Total successful free throws made by the team.

FTA: Free Throws Attempted - Total free throws attempted by the team.

FT_PCT: Free Throw Percentage - The ratio of FTM to FTA expressed as a percentage.

OREB: Offensive Rebounds - Total offensive rebounds collected by the team.

DREB: Defensive Rebounds - Total defensive rebounds collected by the team.

REB: Total Rebounds - Sum of offensive and defensive rebounds (OREB + DREB).

AST: Assists - Total number of assists made by the team.

STL: Steals - Total number of steals made by the team.

BLK: Blocks - Total number of blocks made by the team.

TOV: Turnovers - Total number of turnovers committed by the team.

PF: Personal Fouls - Total number of personal fouls committed by the team.

PLUS_MINUS: Plus/Minus - The point differential when the team is on the court. Positive if the team outscored their opponent during their time on the court, negative if outscored.

## LET'S EXPLORE SOME FACTORS ABOUT TEAM IN MATCHES THAT IMPACT TO THE RESULT (WIN OR LOSE)

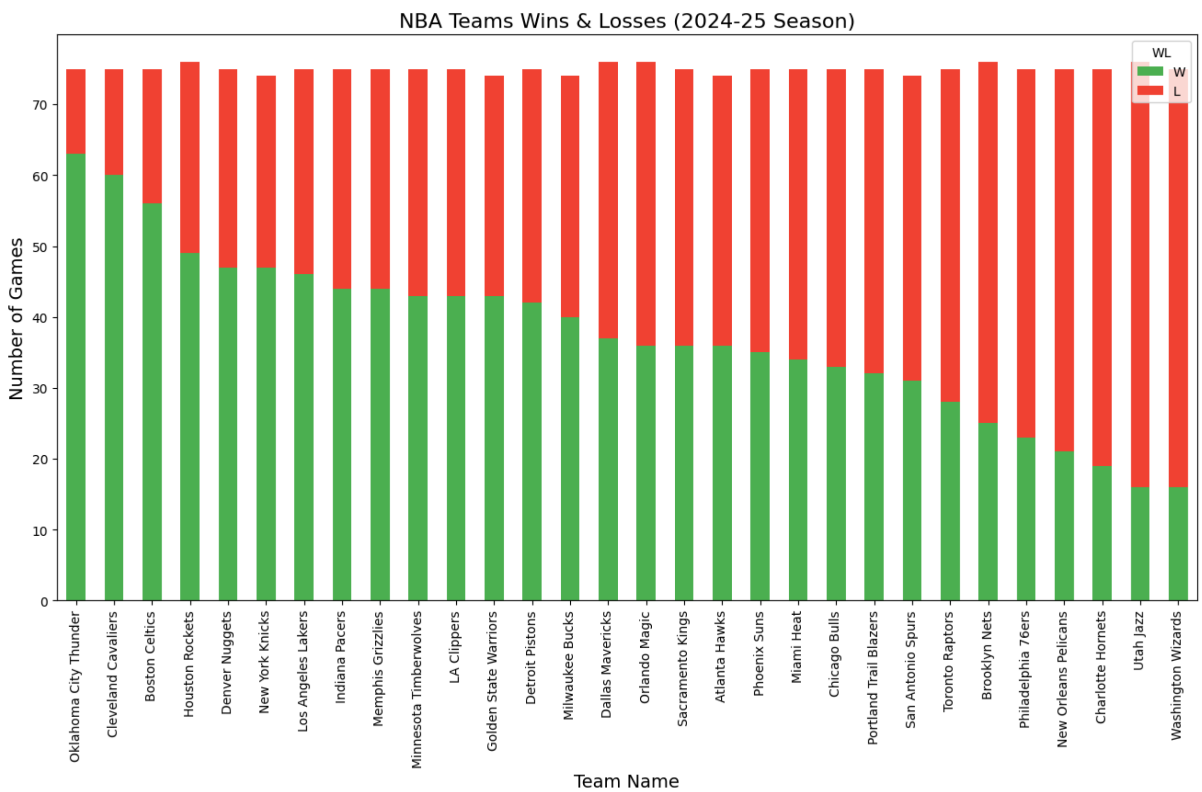
#### THE OVERALL RESULT OF NBA TEAMS (W & L)

In [24]:
```
team_results = df_schedule.groupby(['TEAM_NAME', 'WL']).size().unstack(f

# Add a column for total games played
team_results['Total'] = team_results['W'] + team_results['L']

# Add a column for Win Percentage
team_results['Win%'] = (team_results['W'] / team_results['Total']) * 100

# Display the result
print(team_results)
```

```
WL                       L   W  Total       Win%
TEAM_NAME
Atlanta Hawks           38  36     74  48.648649
Boston Celtics          19  56     75  74.666667
Brooklyn Nets           51  25     76  32.894737
Charlotte Hornets       56  19     75  25.333333
Chicago Bulls           42  33     75  44.000000
Cleveland Cavaliers     15  60     75  80.000000
Dallas Mavericks        39  37     76  48.684211
Denver Nuggets          28  47     75  62.666667
Detroit Pistons         33  42     75  56.000000
Golden State Warriors   31  43     74  58.108108
Houston Rockets         27  49     76  64.473684
Indiana Pacers          31  44     75  58.666667
LA Clippers             32  43     75  57.333333
Los Angeles Lakers      29  46     75  61.333333
Memphis Grizzlies       31  44     75  58.666667
Miami Heat              41  34     75  45.333333
Milwaukee Bucks         34  40     74  54.054054
Minnesota Timberwolves  32  43     75  57.333333
New Orleans Pelicans    54  21     75  28.000000
New York Knicks         27  47     74  63.513514
Oklahoma City Thunder   12  63     75  84.000000
Orlando Magic           40  36     76  47.368421
Philadelphia 76ers      52  23     75  30.666667
Phoenix Suns            40  35     75  46.666667
Portland Trail Blazers  43  32     75  42.666667
Sacramento Kings        39  36     75  48.000000
San Antonio Spurs       43  31     74  41.891892
Toronto Raptors         47  28     75  37.333333
Utah Jazz               60  16     76  21.052632
Washington Wizards      59  16     75  21.333333
```

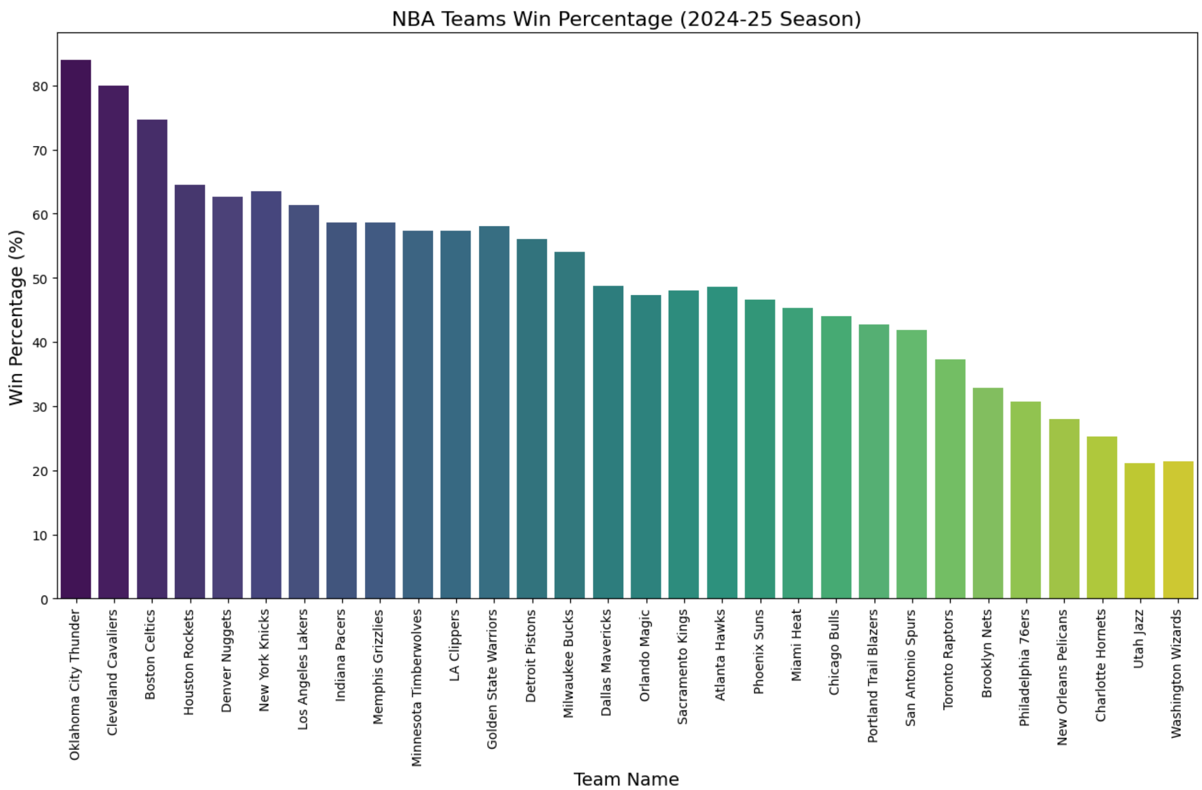
In [25]:
```
plt.figure(figsize=(16, 8))

# Plot Wins & Losses side by side for each team
team_results[['W', 'L']].sort_values(by='W', ascending=False).plot(kind=

plt.title('NBA Teams Wins & Losses (2024–25 Season)', fontsize=16)
plt.xlabel('Team Name', fontsize=14)
plt.ylabel('Number of Games', fontsize=14)
plt.xticks(rotation=90)
plt.show()
```

```
<Figure size 1600x800 with 0 Axes>
```



In [28]:
```
# Sort teams by Wins (W) from highest to lowest
team_results = team_results.sort_values(by='W', ascending=False)

plt.figure(figsize=(16, 8))
sns.barplot(x=team_results.index, y=team_results['Win%'], palette='virid
plt.xticks(rotation=90)
plt.title('NBA Teams Win Percentage (2024-25 Season)', fontsize=16)
plt.xlabel('Team Name', fontsize=14)
plt.ylabel('Win Percentage (%)', fontsize=14)
plt.show()
```

```
/var/folders/5n/r6b07v4s5cs3379jq_dtb8q00000gn/T/ipykernel_8135/1984061
364.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be rem
oved in v0.14.0. Assign the `x` variable to `hue` and set `legend=False
` for the same effect.

  sns.barplot(x=team_results.index, y=team_results['Win%'], palette='vi
ridis')
```



### OKC, CAV and CELTIC got the most winning percentage (>= 75), 10 next teams got the percentage of win around 50-75, the last half got under 50%, noticeable that Jazz and Washington Wizard got under 25%

#### LET'S DO SOME COMPARISION BETWEEN WIN PERCENTAGE AND SOME OTHER STAT

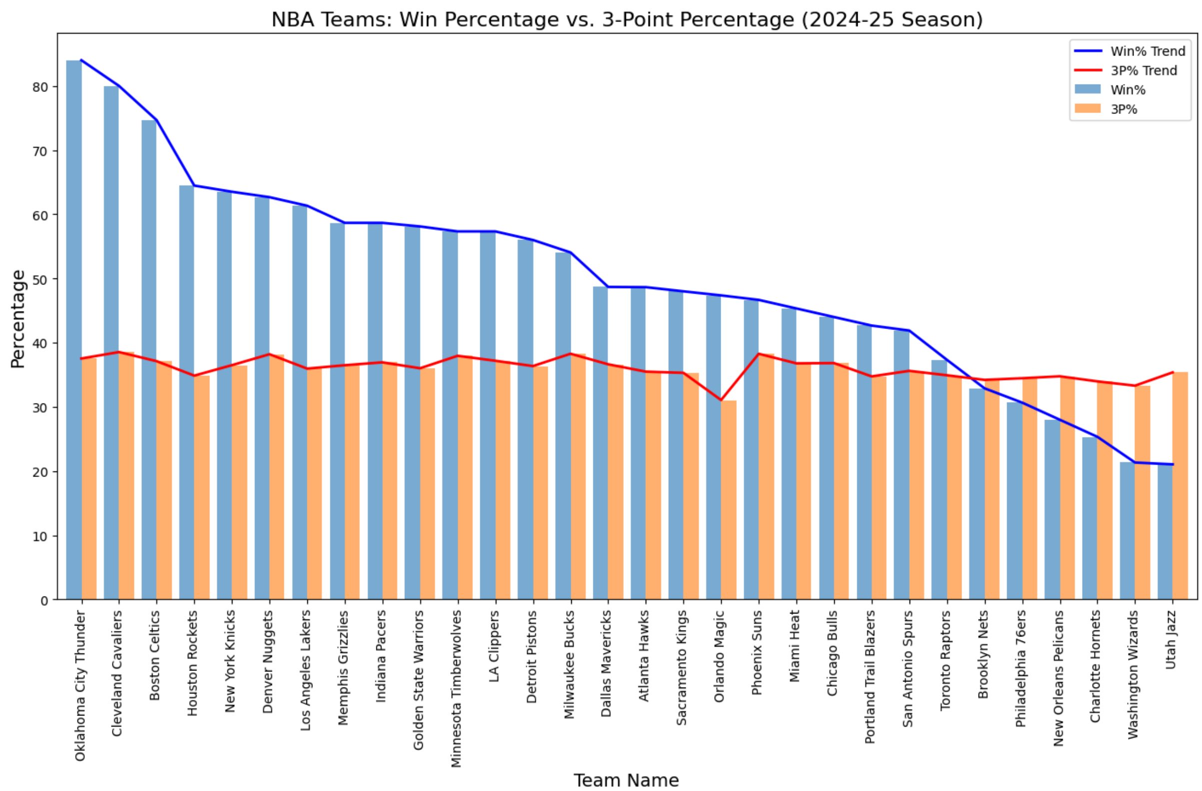
#### is it true that, team that got good 3pts shooters has higher chance to win?

In [30]:
```
#lets compare the win percentage of the teams and the percentage of 3 po
team_3p_pct = df_schedule.groupby('TEAM_NAME')['FG3_PCT'].mean() * 100

# Combine both metrics into a single DataFrame
team_comparison = pd.concat([team_results['Win%'], team_3p_pct], axis=1)
team_comparison.columns = ['Win%', '3P%']

# Sort by Win Percentage
team_comparison_sorted = team_comparison.sort_values(by='Win%', ascendin
```

In [34]:

```
plt.figure(figsize=(16, 8))

# Plotting bar plots for 'Win%' and '3P%'
team_comparison_sorted[['Win%', '3P%']].plot(kind='bar', figsize=(16, 8)

# Adding trend lines for 'Win%' and '3P%'
sns.lineplot(x=np.arange(len(team_comparison_sorted)), y=team_comparison
sns.lineplot(x=np.arange(len(team_comparison_sorted)), y=team_comparison

plt.title('NBA Teams: Win Percentage vs. 3-Point Percentage (2024-25 Sea
plt.xlabel('Team Name', fontsize=14)
plt.ylabel('Percentage', fontsize=14)
plt.xticks(ticks=np.arange(len(team_comparison_sorted)), labels=team_com
plt.legend(loc='upper right')
plt.show()
```

<Figure size 1600x800 with 0 Axes>

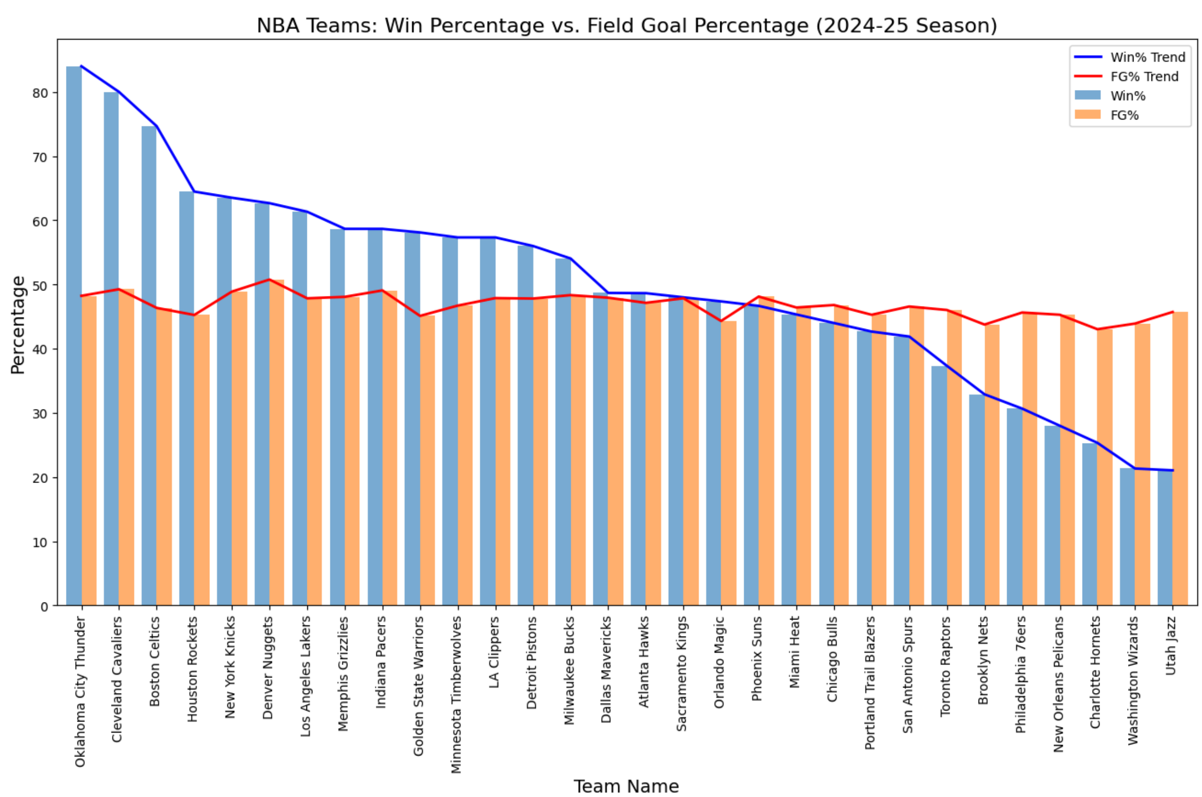



**AS OBSERVE, IT IS NOT TRUE THAT TEAMS GOT MORE ACCURACY OF 3PTS TENDS TO WIN**

In [35]:
```
#how about overall shooting accuracy and win percentage
team_fg_pct = df_schedule.groupby('TEAM_NAME')['FG_PCT'].mean() * 100  #
team_comparison = pd.concat([team_results['Win%'], team_fg_pct], axis=1)
team_comparison.columns = ['Win%', 'FG%']
team_comparison_fg = team_comparison.sort_values(by='Win%', ascending=Fa
```

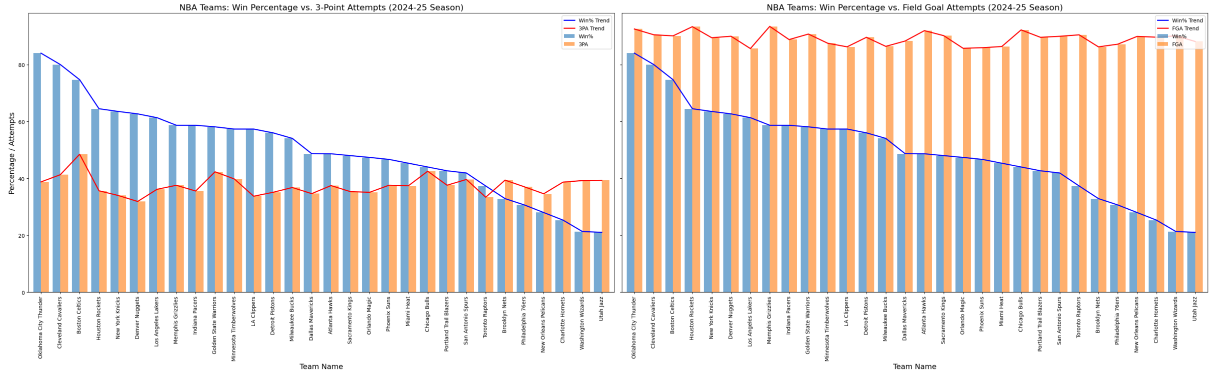
In [37]:
```

plt.figure(figsize=(16, 8))

team_comparison_fg[['Win%', 'FG%']].plot(kind='bar', figsize=(16, 8), wi

# Adding trend lines for 'Win%' and '3P%'
sns.lineplot(x=np.arange(len(team_comparison_fg)), y=team_comparison_fg[
sns.lineplot(x=np.arange(len(team_comparison_fg)), y=team_comparison_fg[

plt.title('NBA Teams: Win Percentage vs. Field Goal Percentage (2024-25 
plt.xlabel('Team Name', fontsize=14)
plt.ylabel('Percentage', fontsize=14)
plt.xticks(ticks=np.arange(len(team_comparison_sorted)), labels=team_com
plt.legend(loc='upper right')
plt.show()
```

```
<Figure size 1600x800 with 0 Axes>
```

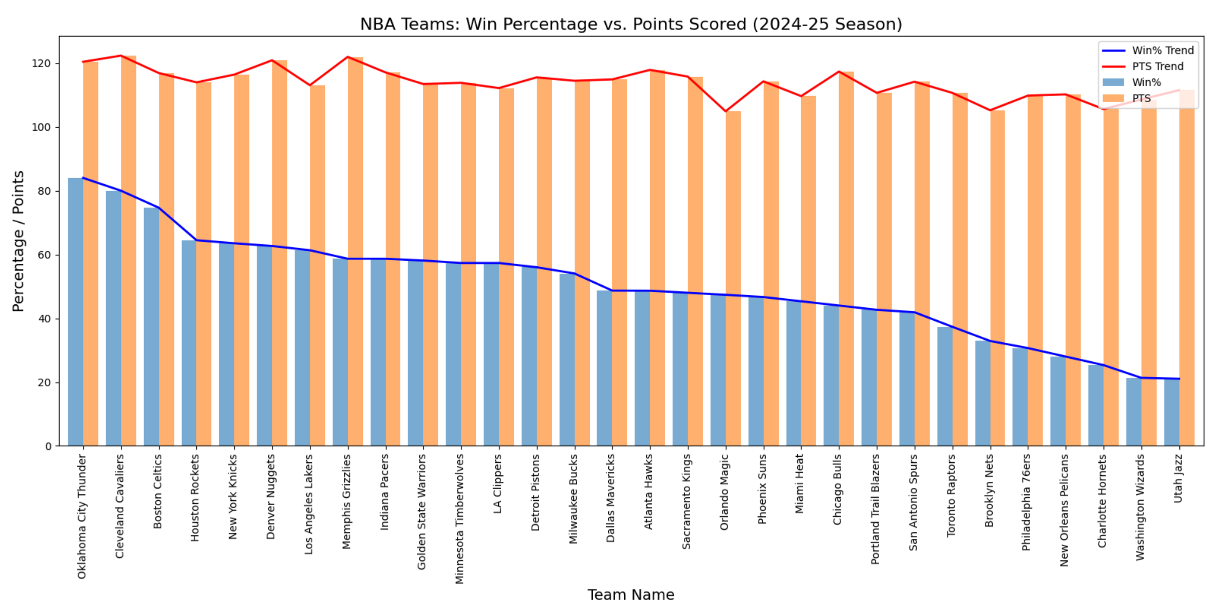


## AGAIN, FIELD GOAL PERCENTAGE SAYS NOTHING TO THE WIN % (WE CAN SEE THAT THE QUALITY OF PLAYER IS QUITE NEAR FOR EVERY TEAM FOR OFFENSIVE STAT)

In [38]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Calculate average 3PA and FGA for each team
team_3pa = df_schedule.groupby('TEAM_NAME')['FG3A'].mean()
team_fga = df_schedule.groupby('TEAM_NAME')['FGA'].mean()

# Combine Win%, 3PA, and FGA
team_comparison_3pa = pd.concat([team_results['Win%'], team_3pa], axis=1
team_comparison_3pa.columns = ['Win%', '3PA']
team_comparison_3pa = team_comparison_3pa.sort_values(by='Win%', ascendi

team_comparison_fga = pd.concat([team_results['Win%'], team_fga], axis=1
team_comparison_fga.columns = ['Win%', 'FGA']
team_comparison_fga = team_comparison_fga.sort_values(by='Win%', ascendi

# Create subplots
fig, axes = plt.subplots(1, 2, figsize=(32, 10), sharey=True)

# Plot for 3PA
team_comparison_3pa[['Win%', '3PA']].plot(kind='bar', ax=axes[0], width=
sns.lineplot(x=np.arange(len(team_comparison_3pa)), y=team_comparison_3p
sns.lineplot(x=np.arange(len(team_comparison_3pa)), y=team_comparison_3p

axes[0].set_title('NBA Teams: Win Percentage vs. 3-Point Attempts (2024-
axes[0].set_xlabel('Team Name', fontsize=14)
axes[0].set_ylabel('Percentage / Attempts', fontsize=14)
axes[0].set_xticks(np.arange(len(team_comparison_3pa)))
axes[0].set_xticklabels(team_comparison_3pa.index, rotation=90)
axes[0].legend(loc='upper right')

# Plot for FGA
team_comparison_fga[['Win%', 'FGA']].plot(kind='bar', ax=axes[1], width=
sns.lineplot(x=np.arange(len(team_comparison_fga)), y=team_comparison_fg
sns.lineplot(x=np.arange(len(team_comparison_fga)), y=team_comparison_fg

axes[1].set_title('NBA Teams: Win Percentage vs. Field Goal Attempts (20
axes[1].set_xlabel('Team Name', fontsize=14)
axes[1].set_ylabel('Percentage / Attempts', fontsize=14)
axes[1].set_xticks(np.arange(len(team_comparison_fga)))
axes[1].set_xticklabels(team_comparison_fga.index, rotation=90)
axes[1].legend(loc='upper right')

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


NBA Teams: Win Percentage vs. 3-Point Attempts (2024-25 Season)
Win% Trend
3PA Trend
Win%
3PA
Percentage / Attempts
0
20
40
60
80
Oklahoma City Thunder
Cleveland Cavaliers
Boston Celtics
Houston Rockets
New York Knicks
Denver Nuggets
Los Angeles Lakers
Memphis Grizzlies
Indiana Pacers
Golden State Warriors
Minnesota Timberwolves
LA Clippers
Detroit Pistons
Milwaukee Bucks
Dallas Mavericks
Atlanta Hawks
Sacramento Kings
Orlando Magic
Phoenix Suns
Miami Heat
Chicago Bulls
Portland Trail Blazers
San Antonio Spurs
Toronto Raptors
Brooklyn Nets
Philadelphia 76ers
New Orleans Pelicans
Charlotte Hornets
Washington Wizards
Utah Jazz
Team Name
NBA Teams: Win Percentage vs. Field Goal Attempts (2024-25 Season)
Win% Trend
FGA Trend
Win%
FGA
Oklahoma City Thunder
Cleveland Cavaliers
Boston Celtics
Houston Rockets
New York Knicks
Denver Nuggets
Los Angeles Lakers
Memphis Grizzlies
Indiana Pacers
Golden State Warriors
Minnesota Timberwolves
LA Clippers
Detroit Pistons
Milwaukee Bucks
Dallas Mavericks
Atlanta Hawks
Sacramento Kings
Orlando Magic
Phoenix Suns
Miami Heat
Chicago Bulls
Portland Trail Blazers
San Antonio Spurs
Toronto Raptors
Brooklyn Nets
Philadelphia 76ers
New Orleans Pelicans
Charlotte Hornets
Washington Wizards
Utah Jazz
Team Name

In [40]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Calculate average PTS for each team
team_pts = df_schedule.groupby('TEAM_NAME')['PTS'].mean()

# Combine Win% and PTS
team_comparison_pts = pd.concat([team_results['Win%'], team_pts], axis=1
team_comparison_pts.columns = ['Win%', 'PTS']
team_comparison_pts = team_comparison_pts.sort_values(by='Win%', ascendi

# Plot
fig, ax = plt.subplots(figsize=(16, 8))

# Plot bar chart
team_comparison_pts[['Win%', 'PTS']].plot(kind='bar', ax=ax, width=0.8, 

# Adding trend lines
sns.lineplot(x=np.arange(len(team_comparison_pts)), y=team_comparison_pt
sns.lineplot(x=np.arange(len(team_comparison_pts)), y=team_comparison_pt

# Formatting the plot
ax.set_title('NBA Teams: Win Percentage vs. Points Scored (2024-25 Seaso
ax.set_xlabel('Team Name', fontsize=14)
ax.set_ylabel('Percentage / Points', fontsize=14)
ax.set_xticks(np.arange(len(team_comparison_pts)))
ax.set_xticklabels(team_comparison_pts.index, rotation=90)
ax.legend(loc='upper right')

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



## CONCLUSTION: OFENSIVE STAT SAYS NOTHING ABOUT IF A TEAM HAS HIGH PERCENTAGE OF WIN

📌 Why PTS May Not Correlate Well With Win%: Pace of Play: Teams that play at a faster pace will generally score more points but may not be more efficient or successful.

Offensive Efficiency: Just scoring a lot of points doesn't guarantee wins if the team also allows a lot of points.

Defensive Performance: Teams with strong defenses can win games even if their scoring is average.

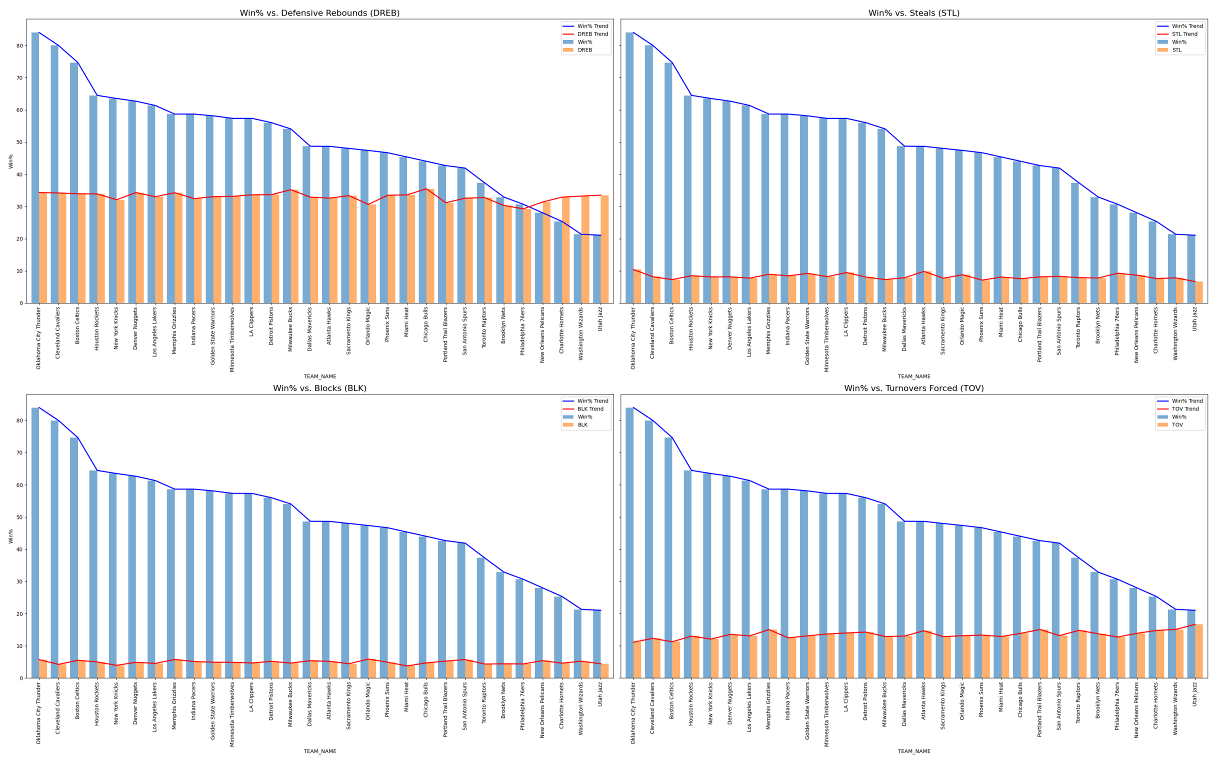
Consistency: High variance in scoring (good games vs. bad games) may make PTS a poor predictor of win percentage.

Close Games: Winning close games may depend more on clutch performance or defense than on overall scoring ability.

# LETS COMPARE DEFENSIVE STAT AND WIN PERCENTAGE

In [41]:
```
# Calculate average Defensive Metrics for each team
team_dreb = df_schedule.groupby('TEAM_NAME')['DREB'].mean()
team_stl = df_schedule.groupby('TEAM_NAME')['STL'].mean()
team_blk = df_schedule.groupby('TEAM_NAME')['BLK'].mean()
team_tov = df_schedule.groupby('TEAM_NAME')['TOV'].mean()

# Combine Win% with each defensive metric
team_comparison_dreb = pd.concat([team_results['Win%'], team_dreb], axis
team_comparison_dreb.columns = ['Win%', 'DREB']

team_comparison_stl = pd.concat([team_results['Win%'], team_stl], axis=1
team_comparison_stl.columns = ['Win%', 'STL']

team_comparison_blk = pd.concat([team_results['Win%'], team_blk], axis=1
team_comparison_blk.columns = ['Win%', 'BLK']

team_comparison_tov = pd.concat([team_results['Win%'], team_tov], axis=1
team_comparison_tov.columns = ['Win%', 'TOV']

# Create subplots
fig, axes = plt.subplots(2, 2, figsize=(32, 20), sharey=True)

# Plot for DREB
team_comparison_dreb[['Win%', 'DREB']].plot(kind='bar', ax=axes[0, 0], w
sns.lineplot(x=np.arange(len(team_comparison_dreb)), y=team_comparison_d
sns.lineplot(x=np.arange(len(team_comparison_dreb)), y=team_comparison_d
axes[0, 0].set_title('Win% vs. Defensive Rebounds (DREB)', fontsize=16)
axes[0, 0].set_xticks(np.arange(len(team_comparison_dreb)))
axes[0, 0].set_xticklabels(team_comparison_dreb.index, rotation=90)
axes[0, 0].legend(loc='upper right')

# Plot for STL
team_comparison_stl[['Win%', 'STL']].plot(kind='bar', ax=axes[0, 1], wid
sns.lineplot(x=np.arange(len(team_comparison_stl)), y=team_comparison_st
sns.lineplot(x=np.arange(len(team_comparison_stl)), y=team_comparison_st
axes[0, 1].set_title('Win% vs. Steals (STL)', fontsize=16)
axes[0, 1].set_xticks(np.arange(len(team_comparison_stl)))
axes[0, 1].set_xticklabels(team_comparison_stl.index, rotation=90)
axes[0, 1].legend(loc='upper right')

# Plot for BLK
team_comparison_blk[['Win%', 'BLK']].plot(kind='bar', ax=axes[1, 0], wid
sns.lineplot(x=np.arange(len(team_comparison_blk)), y=team_comparison_bl
sns.lineplot(x=np.arange(len(team_comparison_blk)), y=team_comparison_bl
axes[1, 0].set_title('Win% vs. Blocks (BLK)', fontsize=16)
axes[1, 0].set_xticks(np.arange(len(team_comparison_blk)))
axes[1, 0].set_xticklabels(team_comparison_blk.index, rotation=90)
axes[1, 0].legend(loc='upper right')

# Plot for TOV
team_comparison_tov[['Win%', 'TOV']].plot(kind='bar', ax=axes[1, 1], wid
sns.lineplot(x=np.arange(len(team_comparison_tov)), y=team_comparison_to
sns.lineplot(x=np.arange(len(team_comparison_tov)), y=team_comparison_to
axes[1, 1].set_title('Win% vs. Turnovers Forced (TOV)', fontsize=16)
axes[1, 1].set_xticks(np.arange(len(team_comparison_tov)))
axes[1, 1].set_xticklabels(team_comparison_tov.index, rotation=90)
axes[1, 1].legend(loc='upper right')
```

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




DREB (Defensive Rebounds):

It has some variation, but the trend line is mostly flat compared to Win%.

Teams with high Win% don't necessarily have high DREB.

STL (Steals):

The trend is generally low across all teams.

Steals don't appear to correlate strongly with Win%.

BLK (Blocks):

The trend is also quite flat.

Blocks are a rare event in games, so they may not have a large impact on overall team success.

TOV (Turnovers Forced):

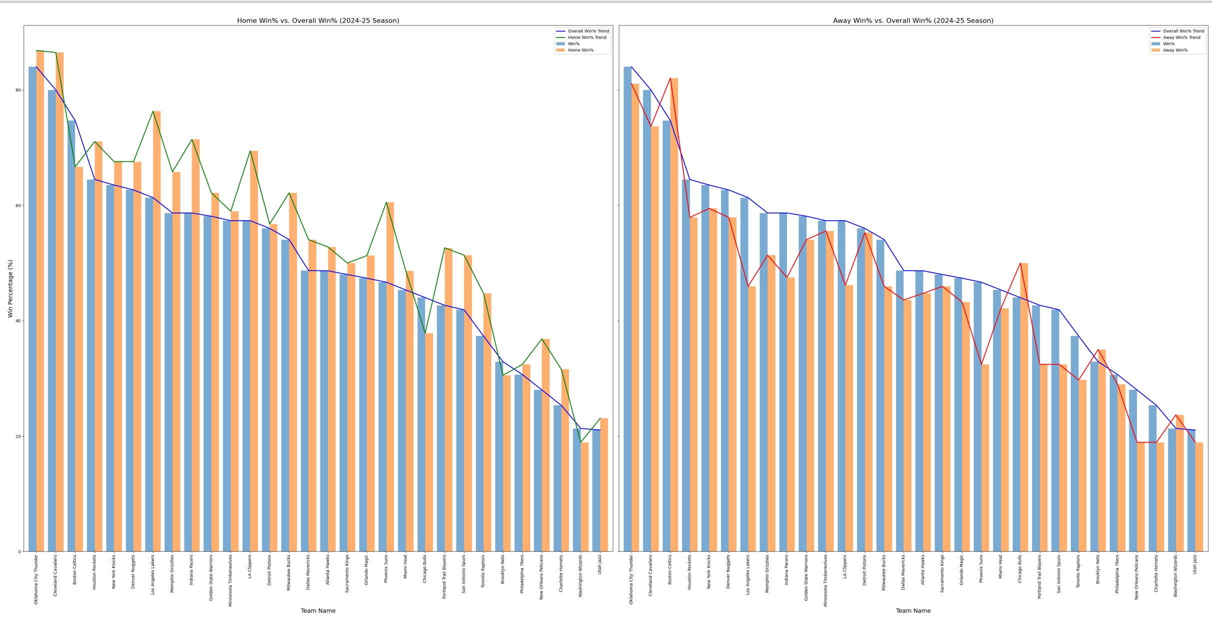
Similar to STL, this trend is flat.

Forcing turnovers might be important but doesn’t seem directly related to Win%.

In [46]:
```
# Filter home and away games
home_games = df_schedule[df_schedule['MATCHUP'].str.contains('vs.')]
away_games = df_schedule[df_schedule['MATCHUP'].str.contains('@')]

# Calculate total home games and home wins for each team
home_results = home_games.groupby(['TEAM_NAME', 'WL']).size().unstack(fi
home_results['Total Home Games'] = home_results['W'] + home_results['L']
home_results['Home Win%'] = (home_results['W'] / home_results['Total Hom

# Calculate total away games and away wins for each team
away_results = away_games.groupby(['TEAM_NAME', 'WL']).size().unstack(fi
away_results['Total Away Games'] = away_results['W'] + away_results['L']
away_results['Away Win%'] = (away_results['W'] / away_results['Total Awa

# Merge Overall Win%, Home Win%, and Away Win%
team_comparison = pd.concat([team_results['Win%'], home_results['Home Wi
team_comparison = team_comparison.sort_values(by='Win%', ascending=False

# Display the merged data
print(team_comparison)
```

```
                             Win%  Home Win%  Away Win%
TEAM_NAME
Oklahoma City Thunder   84.000000  86.842105  81.081081
Cleveland Cavaliers     80.000000  86.486486  73.684211
Boston Celtics          74.666667  66.666667  82.051282
Houston Rockets         64.473684  71.052632  57.894737
New York Knicks         63.513514  67.567568  59.459459
Denver Nuggets          62.666667  67.567568  57.894737
Los Angeles Lakers      61.333333  76.315789  45.945946
Memphis Grizzlies       58.666667  65.789474  51.351351
Indiana Pacers          58.666667  71.428571  47.500000
Golden State Warriors   58.108108  62.162162  54.054054
Minnesota Timberwolves  57.333333  58.974359  55.555556
LA Clippers             57.333333  69.444444  46.153846
Detroit Pistons         56.000000  56.756757  55.263158
Milwaukee Bucks         54.054054  62.162162  45.945946
Dallas Mavericks        48.684211  54.054054  43.589744
Atlanta Hawks           48.648649  52.777778  44.736842
Sacramento Kings        48.000000  50.000000  45.945946
Orlando Magic           47.368421  51.282051  43.243243
Phoenix Suns            46.666667  60.526316  32.432432
Miami Heat              45.333333  48.648649  42.105263
Chicago Bulls           44.000000  37.837838  50.000000
Portland Trail Blazers  42.666667  52.631579  32.432432
San Antonio Spurs       41.891892  51.351351  32.432432
Toronto Raptors         37.333333  44.736842  29.729730
Brooklyn Nets           32.894737  30.555556  35.000000
Philadelphia 76ers      30.666667  32.432432  28.947368
New Orleans Pelicans    28.000000  36.842105  18.918919
Charlotte Hornets       25.333333  31.578947  18.918919
Washington Wizards      21.333333  18.918919  23.684211
Utah Jazz               21.052632  23.076923  18.918919
```

In [48]:
```
# Create subplots
fig, axes = plt.subplots(1, 2, figsize=(40, 20), sharey=True)

# Plot Home Win% vs Overall Win%
team_comparison[['Win%', 'Home Win%']].plot(kind='bar', ax=axes[0], widt
sns.lineplot(x=np.arange(len(team_comparison)), y=team_comparison['Win%'
sns.lineplot(x=np.arange(len(team_comparison)), y=team_comparison['Home \

axes[0].set_title('Home Win% vs. Overall Win% (2024-25 Season)', fontsiz
axes[0].set_xlabel('Team Name', fontsize=14)
axes[0].set_ylabel('Win Percentage (%)', fontsize=14)
axes[0].set_xticks(np.arange(len(team_comparison)))
axes[0].set_xticklabels(team_comparison.index, rotation=90)
axes[0].legend(loc='upper right')

# Plot Away Win% vs Overall Win%
team_comparison[['Win%', 'Away Win%']].plot(kind='bar', ax=axes[1], widt
sns.lineplot(x=np.arange(len(team_comparison)), y=team_comparison['Win%'
sns.lineplot(x=np.arange(len(team_comparison)), y=team_comparison['Away \

axes[1].set_title('Away Win% vs. Overall Win% (2024-25 Season)', fontsiz
axes[1].set_xlabel('Team Name', fontsize=14)
axes[1].set_ylabel('Win Percentage (%)', fontsize=14)
axes[1].set_xticks(np.arange(len(team_comparison)))
axes[1].set_xticklabels(team_comparison.index, rotation=90)
axes[1].legend(loc='upper right')

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




Home Win% vs. Overall Win% (Left Plot):

Teams with high Overall Win% generally have even higher Home Win%.

Noticeable peaks in Home Win% suggest that certain teams are particularly strong at home (e.g., Oklahoma City Thunder, Cleveland Cavaliers).

The green trend line (Home Win%) is generally above the blue line (Overall Win%).

Away Win% vs. Overall Win% (Right Plot):

Teams with high Overall Win% tend to maintain solid Away Win%, but it's typically lower than their Home Win%.

Noticeable dips in the red trend line (Away Win%) indicate teams that struggle more on the road.

The red trend line is mostly below the blue line (Overall Win%).

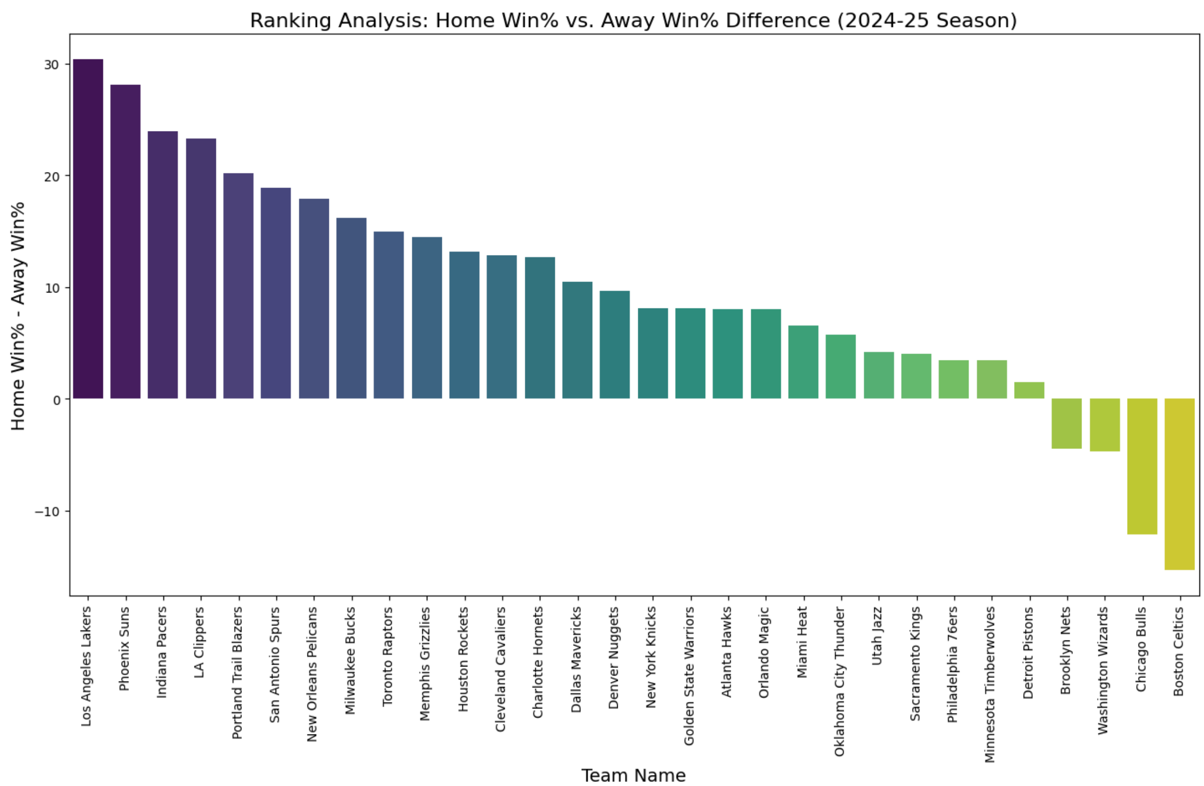
Some teams in the middle for example LA Lakers or LA CLiper got high %win at home but low at away, and in contrast is Boston Celtic (good at away but bad at home) (good and bad here is comparing with overall win and the other (away vs home) not comparing to other team)

In [49]:
```
# Calculate the difference between Home Win% and Away Win%
team_comparison['Home vs. Away Difference'] = team_comparison['Home Win%

# Rank teams by the difference between Home Win% and Away Win%
ranked_teams = team_comparison.sort_values(by='Home vs. Away Difference'

# Display the ranked teams
print(ranked_teams[['Home Win%', 'Away Win%', 'Home vs. Away Difference'
```

```
                         Home Win%  Away Win%  Home vs. Away Difference
TEAM_NAME
Los Angeles Lakers       76.315789  45.945946                 30.369844
Phoenix Suns             60.526316  32.432432                 28.093883
Indiana Pacers           71.428571  47.500000                 23.928571
LA Clippers              69.444444  46.153846                 23.290598
Portland Trail Blazers   52.631579  32.432432                 20.199147
San Antonio Spurs        51.351351  32.432432                 18.918919
New Orleans Pelicans     36.842105  18.918919                 17.923186
Milwaukee Bucks          62.162162  45.945946                 16.216216
Toronto Raptors          44.736842  29.729730                 15.007112
Memphis Grizzlies        65.789474  51.351351                 14.438122
Houston Rockets          71.052632  57.894737                 13.157895
Cleveland Cavaliers      86.486486  73.684211                 12.802276
Charlotte Hornets        31.578947  18.918919                 12.660028
Dallas Mavericks         54.054054  43.589744                 10.464310
Denver Nuggets           67.567568  57.894737                  9.672831
New York Knicks          67.567568  59.459459                  8.108108
Golden State Warriors    62.162162  54.054054                  8.108108
Atlanta Hawks            52.777778  44.736842                  8.040936
Orlando Magic            51.282051  43.243243                  8.038808
Miami Heat               48.648649  42.105263                  6.543385
Oklahoma City Thunder    86.842105  81.081081                  5.761024
Utah Jazz                23.076923  18.918919                  4.158004
Sacramento Kings         50.000000  45.945946                  4.054054
Philadelphia 76ers       32.432432  28.947368                  3.485064
Minnesota Timberwolves   58.974359  55.555556                  3.418803
Detroit Pistons          56.756757  55.263158                  1.493599
Brooklyn Nets            30.555556  35.000000                 -4.444444
Washington Wizards       18.918919  23.684211                 -4.765292
Chicago Bulls            37.837838  50.000000                -12.162162
Boston Celtics           66.666667  82.051282                -15.384615
```

In [50]:
```
plt.figure(figsize=(16, 8))

# Plot bar chart of Home vs. Away Difference
sns.barplot(x=ranked_teams.index, y=ranked_teams['Home vs. Away Differen

plt.title('Ranking Analysis: Home Win% vs. Away Win% Difference (2024-25
plt.xlabel('Team Name', fontsize=14)
plt.ylabel('Home Win% - Away Win%', fontsize=14)
plt.xticks(rotation=90)
plt.show()
```

```
/var/folders/5n/r6b07v4s5cs3379jq_dtb8q00000gn/T/ipykernel_8135/1282995
621.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be rem
oved in v0.14.0. Assign the `x` variable to `hue` and set `legend=False
` for the same effect.

  sns.barplot(x=ranked_teams.index, y=ranked_teams['Home vs. Away Diffe
rence'], palette='viridis')
```




Ranking Analysis Plot (Home vs. Away Win% Difference):

Top Teams:

Teams like Los Angeles Lakers, Phoenix Suns, Indiana Pacers, and LA Clippers have a significant positive difference between Home Win% and Away Win%.

This suggests that they perform much better at home than on the road.

Bottom Teams:

Surprisingly, Boston Celtics, Chicago Bulls, Washington Wizards have a negative difference.

This suggests they perform better away from home or struggle more at home. 📌 Interesting Insights: The Los Angeles Lakers are very strong at home compared to away games. This could be due to crowd support, familiarity with the court, or other home advantages.

Boston Celtics have a negative Home vs. Away difference, despite being one of the top performers overall. This suggests they are a good road team but may not be leveraging home advantage effectively.

Washington Wizards are at the bottom in both Win% and Home vs. Away Difference, suggesting general poor performance across all venues.

### Ranking Analysis Plot (Home vs. Away Win% Difference):

Screenshot 2025-04-01 at 9.44.44 PM.png

In [60]:
```
df_schedule['TEAM_NAME'].unique()
```

Out[60]:
```
array(['Memphis Grizzlies', 'Oklahoma City Thunder', 'Brooklyn Nets',
       'Chicago Bulls', 'Orlando Magic', 'Boston Celtics',
       'Sacramento Kings', 'Los Angeles Lakers', 'Dallas Mavericks',
       'Houston Rockets', 'LA Clippers', 'Indiana Pacers',
       'Washington Wizards', 'Utah Jazz', 'Miami Heat',
       'Charlotte Hornets', 'Toronto Raptors', 'Golden State Warriors',
       'Minnesota Timberwolves', 'Cleveland Cavaliers', 'Phoenix Suns',
       'Detroit Pistons', 'Philadelphia 76ers', 'Milwaukee Bucks',
       'Atlanta Hawks', 'San Antonio Spurs', 'Portland Trail Blazers',
       'New York Knicks', 'New Orleans Pelicans', 'Denver Nuggets'],
      dtype=object)
```

In [100]:
```
team_abbreviation_map = {
    # Western Conference Teams
    'OKC': 'Oklahoma City Thunder',
    'HOU': 'Houston Rockets',
    'DEN': 'Denver Nuggets',
    'LAL': 'Los Angeles Lakers',
    'GSW': 'Golden State Warriors',
    'MEM': 'Memphis Grizzlies',
    'DAL': 'Dallas Mavericks',
    'SAC': 'Sacramento Kings',
    'MIN': 'Minnesota Timberwolves',
    'LAC': 'Los Angeles Clippers',

    # Eastern Conference Teams
    'CLE': 'Cleveland Cavaliers',
    'BOS': 'Boston Celtics',
    'NYK': 'New York Knicks',
    'IND': 'Indiana Pacers',
    'DET': 'Detroit Pistons',
    'MIL': 'Milwaukee Bucks',
    'MIA': 'Miami Heat',
    'CHI': 'Chicago Bulls',
    'ORL': 'Orlando Magic',
    'ATL': 'Atlanta Hawks'
}

# List of teams from the bracket
playoff_teams = [
    # Western Conference Teams
    'Oklahoma City Thunder', 'Houston Rockets', 'Denver Nuggets', 'Los A
    'Golden State Warriors', 'Memphis Grizzlies', 'Dallas Mavericks', 'S
    'Minnesota Timberwolves', 'Los Angeles Clippers',

    # Eastern Conference Teams
    'Cleveland Cavaliers', 'Boston Celtics', 'New York Knicks', 'Indiana
    'Detroit Pistons', 'Milwaukee Bucks', 'Miami Heat', 'Chicago Bulls',
    'Orlando Magic', 'Atlanta Hawks'
]


def extract_opponent(row):
    matchup = row['MATCHUP']
    team_name = row['TEAM_NAME']

    # Extract abbreviation of the opponent team
    if 'vs.' in matchup:
        opponent_abbr = matchup.split('vs. ')[1]
    elif '@' in matchup:
        opponent_abbr = matchup.split('@ ')[1]
    else:
        return None

    # Convert abbreviation to full name using the mapping dictionary
    opponent_name = team_abbreviation_map.get(opponent_abbr, None)

    # Return the opponent name if it's in the playoff team list, otherwi
    if opponent_name in playoff_teams:
```

```
        return opponent_name
    else:
        return None

# Apply the function to create the 'OPPONENT' column
df_schedule['OPPONENT'] = df_schedule.apply(extract_opponent, axis=1)
```

In [125]:

```
western_teams = ['Oklahoma City Thunder', 'Houston Rockets', 'Denver Nug
    'Golden State Warriors', 'Memphis Grizzlies', 'Dallas Mavericks', 'S
    'Minnesota Timberwolves', 'Los Angeles Clippers']
eastern_teams = ['Cleveland Cavaliers', 'Boston Celtics', 'New York Knic
    'Detroit Pistons', 'Milwaukee Bucks', 'Miami Heat', 'Chicago Bulls',
    'Orlando Magic', 'Atlanta Hawks']
```

In [126]:

```
df_schedule.head()
```

Out[126]:

| | SEASON_ID | TEAM_ID | TEAM_ABBREVIATION | TEAM_NAME | GAME_ID | GAME_DATE | MATC |
|---|---|---|---|---|---|---|---|
| **2** | 22024 | 1610612763 | MEM | Memphis Grizzlies | 0022401093 | 2025-03-31 | MEI |
| **3** | 22024 | 1610612760 | OKC | Oklahoma City Thunder | 0022401094 | 2025-03-31 | OK |
| **4** | 22024 | 1610612751 | BKN | Brooklyn Nets | 0022401095 | 2025-03-31 | BI |
| **5** | 22024 | 1610612741 | CHI | Chicago Bulls | 0022401094 | 2025-03-31 | C |
| **6** | 22024 | 1610612753 | ORL | Orlando Magic | 0022401091 | 2025-03-31 | OF |

5 rows × 29 columns

In [127]:

```

# Filter the main dataframe to only include games involving these teams
filtered_df = df_schedule[df_schedule['TEAM_NAME'].isin(western_teams) &
```

In [128]:
```
filtered_df.head()
```

Out[128]:

| | SEASON_ID | TEAM_ID | TEAM_ABBREVIATION | TEAM_NAME | GAME_ID | GAME_DATE | MAT |
|---|---|---|---|---|---|---|---|
| **9** | 22024 | 1610612747 | LAL | Los Angeles Lakers | 0022401096 | 2025-03-31 | L |
| **11** | 22024 | 1610612745 | HOU | Houston Rockets | 0022401096 | 2025-03-31 | H |
| **36** | 22024 | 1610612763 | MEM | Memphis Grizzlies | 0022401078 | 2025-03-29 | MI |
| **46** | 22024 | 1610612747 | LAL | Los Angeles Lakers | 0022401078 | 2025-03-29 | |
| **65** | 22024 | 1610612763 | MEM | Memphis Grizzlies | 0022401064 | 2025-03-27 | M |

5 rows × 29 columns

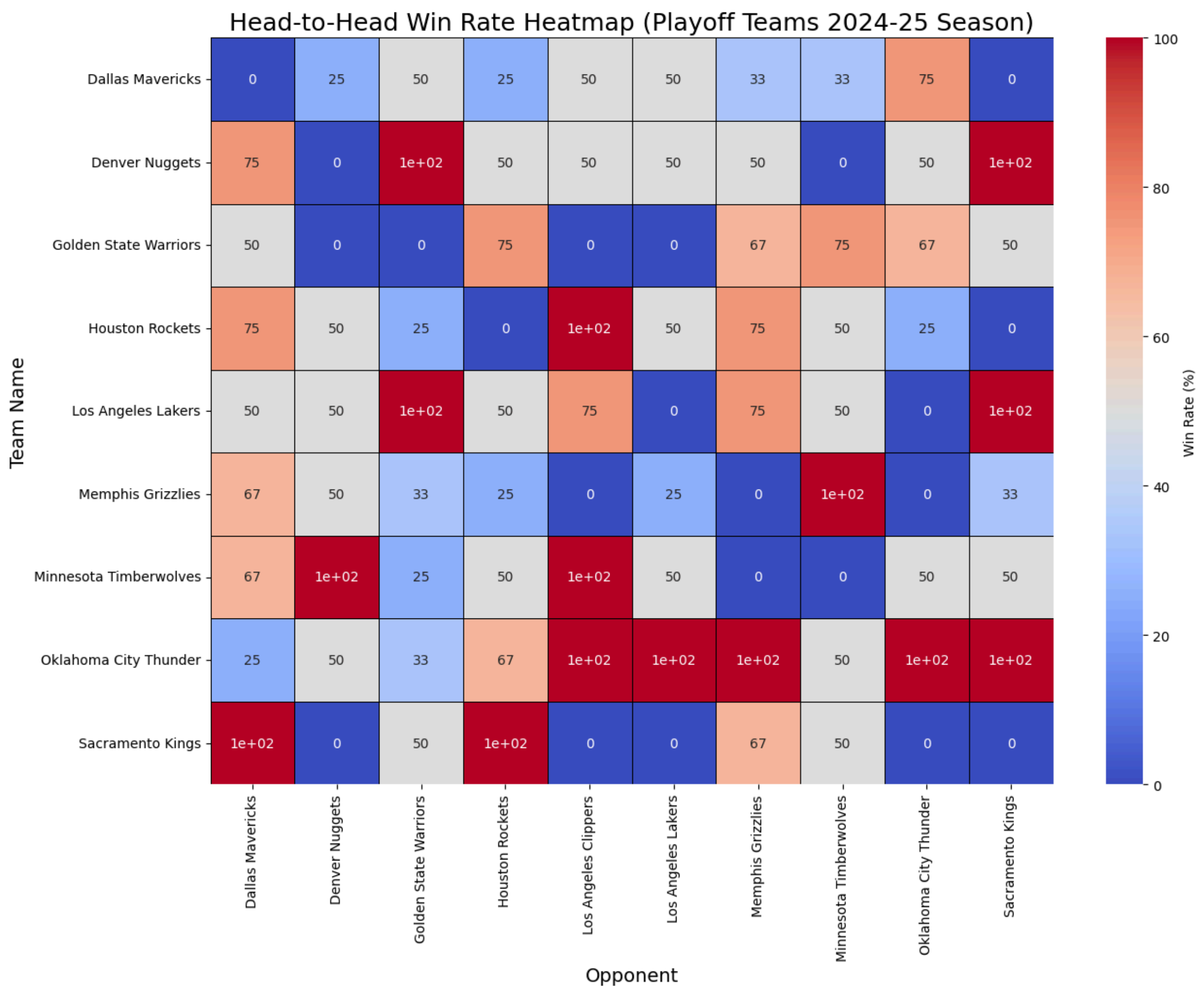
In [129]:
```
filtered_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 265 entries, 9 to 2251
Data columns (total 29 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   SEASON_ID          265 non-null    object
 1   TEAM_ID            265 non-null    int64
 2   TEAM_ABBREVIATION  265 non-null    object
 3   TEAM_NAME          265 non-null    object
 4   GAME_ID            265 non-null    object
 5   GAME_DATE          265 non-null    datetime64[ns]
 6   MATCHUP            265 non-null    object
 7   WL                 265 non-null    object
 8   MIN                265 non-null    int64
 9   PTS                265 non-null    int64
 10  FGM                265 non-null    int64
 11  FGA                265 non-null    int64
 12  FG_PCT             265 non-null    float64
 13  FG3M               265 non-null    int64
 14  FG3A               265 non-null    int64
 15  FG3_PCT            265 non-null    float64
 16  FTM                265 non-null    int64
 17  FTA                265 non-null    int64
 18  FT_PCT             265 non-null    float64
 19  OREB               265 non-null    int64
 20  DREB               265 non-null    int64
 21  REB                265 non-null    int64
 22  AST                265 non-null    int64
 23  STL                265 non-null    int64
 24  BLK                265 non-null    int64
 25  TOV                265 non-null    int64
 26  PF                 265 non-null    int64
 27  PLUS_MINUS         265 non-null    float64
 28  OPPONENT           265 non-null    object
dtypes: datetime64[ns](1), float64(4), int64(17), object(7)
memory usage: 62.1+ KB
```

In [130]:
```
# Calculate win rates between each matchup
matchup_results = filtered_df.groupby(['TEAM_NAME', 'OPPONENT', 'WL']).s

# Pivot the table to have separate columns for Wins and Losses
matchup_pivot = matchup_results.pivot(index=['TEAM_NAME', 'OPPONENT'], c

# Add a Total Games column and Win Rate calculation
matchup_pivot['Total Games'] = matchup_pivot.sum(axis=1)

if 'W' in matchup_pivot.columns:
    matchup_pivot['Win Rate'] = (matchup_pivot['W'] / matchup_pivot['Tot
else:
    matchup_pivot['Win Rate'] = 0  # No wins recorded for that matchup

# Reset the index for better visualization
matchup_pivot.reset_index(inplace=True)
```

## HEAD TO HEAD WIN RATE

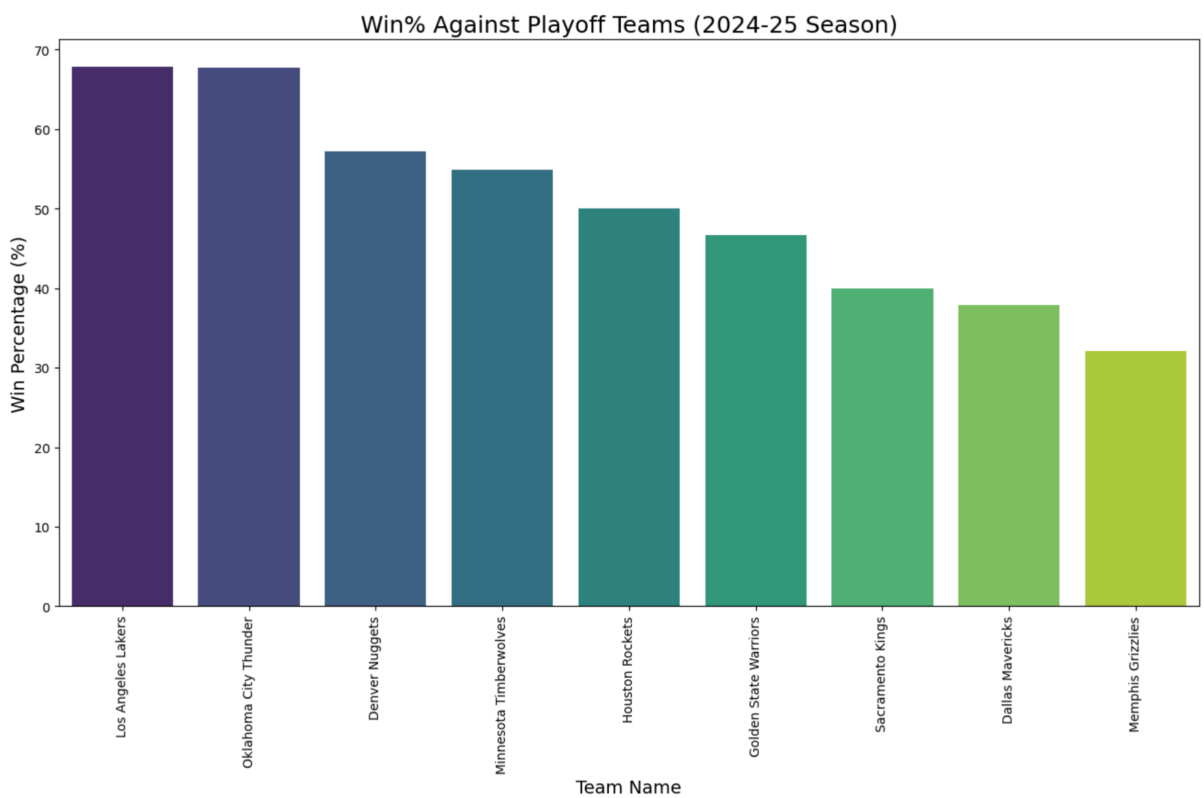
In [131]:
```
heatmap_data = matchup_pivot.pivot_table(index='TEAM_NAME', columns='OPP(
# Plotting the heatmap
plt.figure(figsize=(14, 10))
sns.heatmap(heatmap_data, annot=True, cmap='coolwarm', linewidths=.5, li

plt.title('Head-to-Head Win Rate Heatmap (Playoff Teams 2024-25 Season)'
plt.xlabel('Opponent', fontsize=14)
plt.ylabel('Team Name', fontsize=14)
plt.show()
```



In [132]:
```
# Calculate total games and wins for each team against playoff teams
team_wins = filtered_df[filtered_df['WL'] == 'W'].groupby('TEAM_NAME').s
team_total_games = filtered_df.groupby('TEAM_NAME').size()

# Calculate Win Percentage
playoff_win_percentage = (team_wins / team_total_games) * 100
playoff_win_percentage = playoff_win_percentage.fillna(0).sort_values(as

# Convert to DataFrame for visualization
playoff_win_df = playoff_win_percentage.reset_index()
playoff_win_df.columns = ['TEAM_NAME', 'Win% Against Playoff Teams']
```

In [133]:
```
plt.figure(figsize=(16, 8))
sns.barplot(x='TEAM_NAME', y='Win% Against Playoff Teams', data=playoff_

plt.title('Win% Against Playoff Teams (2024-25 Season)', fontsize=18)
plt.xlabel('Team Name', fontsize=14)
plt.ylabel('Win Percentage (%)', fontsize=14)
plt.xticks(rotation=90)
plt.show()
```

```
/var/folders/5n/r6b07v4s5cs3379jq_dtb8q00000gn/T/ipykernel_8135/4275752
822.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be rem
oved in v0.14.0. Assign the `x` variable to `hue` and set `legend=False
` for the same effect.

  sns.barplot(x='TEAM_NAME', y='Win% Against Playoff Teams', data=playo
ff_win_df, palette='viridis')
```



Top Performers:

Los Angeles Lakers and Oklahoma City Thunder have the highest win percentages against other playoff teams, both around 70%.

This suggests they are strong contenders and can handle tough competition.

Middle Performers:

Denver Nuggets, Minnesota Timberwolves, Houston Rockets, and Golden State Warriors are clustered around the 50%-60% range.

These teams are competitive but not dominant against other top teams.

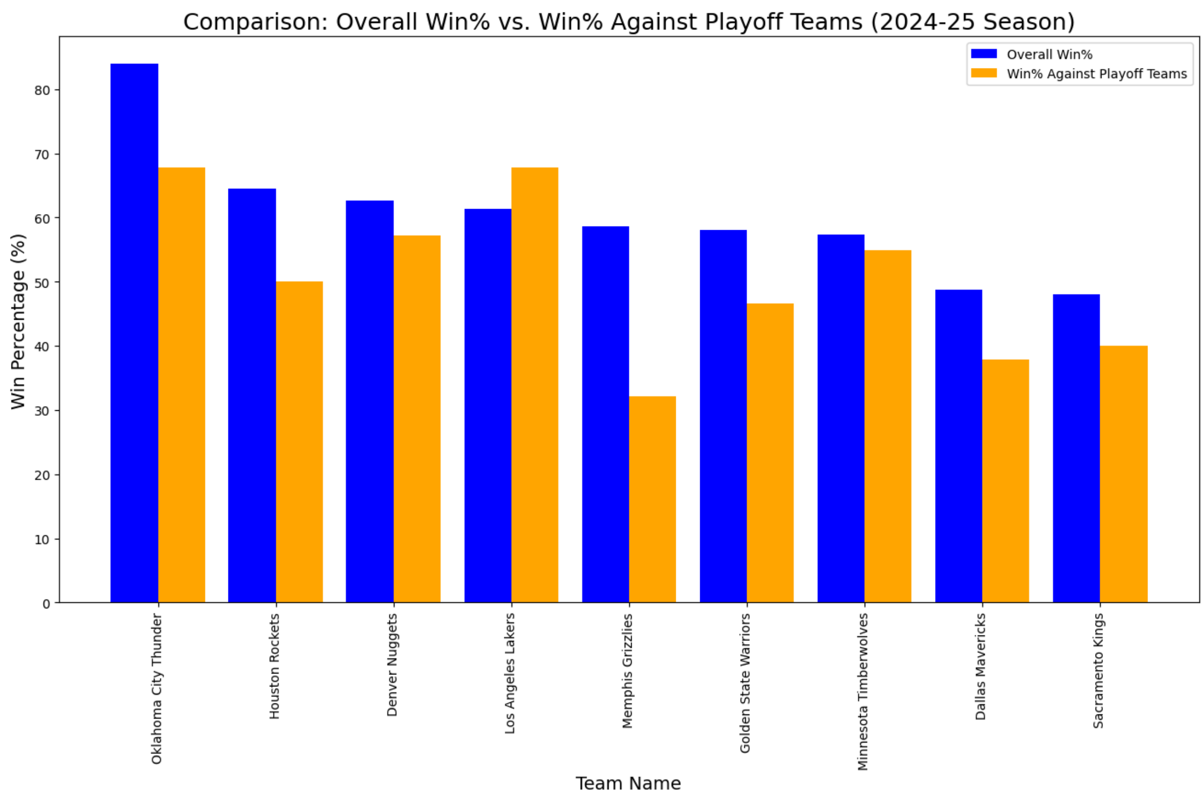
Bottom Performers:

Sacramento Kings, Dallas Mavericks, and Memphis Grizzlies are below 50%, with the Grizzlies having the lowest performance against playoff teams.

This suggests they might be struggling more against stronger opponents.

In [134]:
```
# Calculate total wins and games for all playoff teams
total_wins = df_schedule[(df_schedule['TEAM_NAME'].isin(playoff_teams))
total_games = df_schedule[df_schedule['TEAM_NAME'].isin(playoff_teams)].

# Calculate Overall Win Percentage
overall_win_percentage = (total_wins / total_games) * 100
overall_win_percentage = overall_win_percentage.fillna(0).sort_values(as

# Convert to DataFrame
overall_win_df = overall_win_percentage.reset_index()
overall_win_df.columns = ['TEAM_NAME', 'Overall Win%']

# Merge the overall win percentage with the playoff-only win percentage
comparison_df = pd.merge(playoff_win_df, overall_win_df, on='TEAM_NAME')
```

In [135]:
```
# Sort by Overall Win% for consistency
comparison_df = comparison_df.sort_values(by='Overall Win%', ascending=F

plt.figure(figsize=(16, 8))
bar_width = 0.4
indices = range(len(comparison_df))

# Plotting the two win percentages side by side
plt.bar(indices, comparison_df['Overall Win%'], width=bar_width, label='
plt.bar([i + bar_width for i in indices], comparison_df['Win% Against Pl

plt.title('Comparison: Overall Win% vs. Win% Against Playoff Teams (2024
plt.xlabel('Team Name', fontsize=14)
plt.ylabel('Win Percentage (%)', fontsize=14)
plt.xticks([i + bar_width / 2 for i in indices], comparison_df['TEAM_NAM
plt.legend()
plt.show()
```




INTERESTING INSIGHT

OKC has a great chance of win in all reg and playoff

It is interesting that LA Lakers got middle positon in reg but highest winning rate in playoff

Memphis Grizzlies got good winning rate in overall but bad compare to playoff team

In [136]:
```
df_schedule.head()
```

Out[136]:

| | SEASON_ID | TEAM_ID | TEAM_ABBREVIATION | TEAM_NAME | GAME_ID | GAME_DATE | MATC |
|---|---|---|---|---|---|---|---|
| **2** | 22024 | 1610612763 | MEM | Memphis Grizzlies | 0022401093 | 2025-03-31 | ME |
| **3** | 22024 | 1610612760 | OKC | Oklahoma City Thunder | 0022401094 | 2025-03-31 | OK |
| **4** | 22024 | 1610612751 | BKN | Brooklyn Nets | 0022401095 | 2025-03-31 | BI |
| **5** | 22024 | 1610612741 | CHI | Chicago Bulls | 0022401094 | 2025-03-31 | C |
| **6** | 22024 | 1610612753 | ORL | Orlando Magic | 0022401091 | 2025-03-31 | OF |

5 rows × 29 columns

In [137]:
```
eastern_playoff_teams = [
    'Cleveland Cavaliers', 'Boston Celtics', 'New York Knicks', 'Indiana
    'Detroit Pistons', 'Milwaukee Bucks', 'Miami Heat', 'Chicago Bulls',
    'Orlando Magic', 'Atlanta Hawks'
]

# Filter the main dataframe to only include games involving these teams
eastern_registered = df_schedule[
    (df_schedule['TEAM_NAME'].isin(eastern_playoff_teams)) &
    (df_schedule['OPPONENT'].isin(eastern_playoff_teams))
]
```

In [138]:
```
eastern_registered
```

Out[138]:

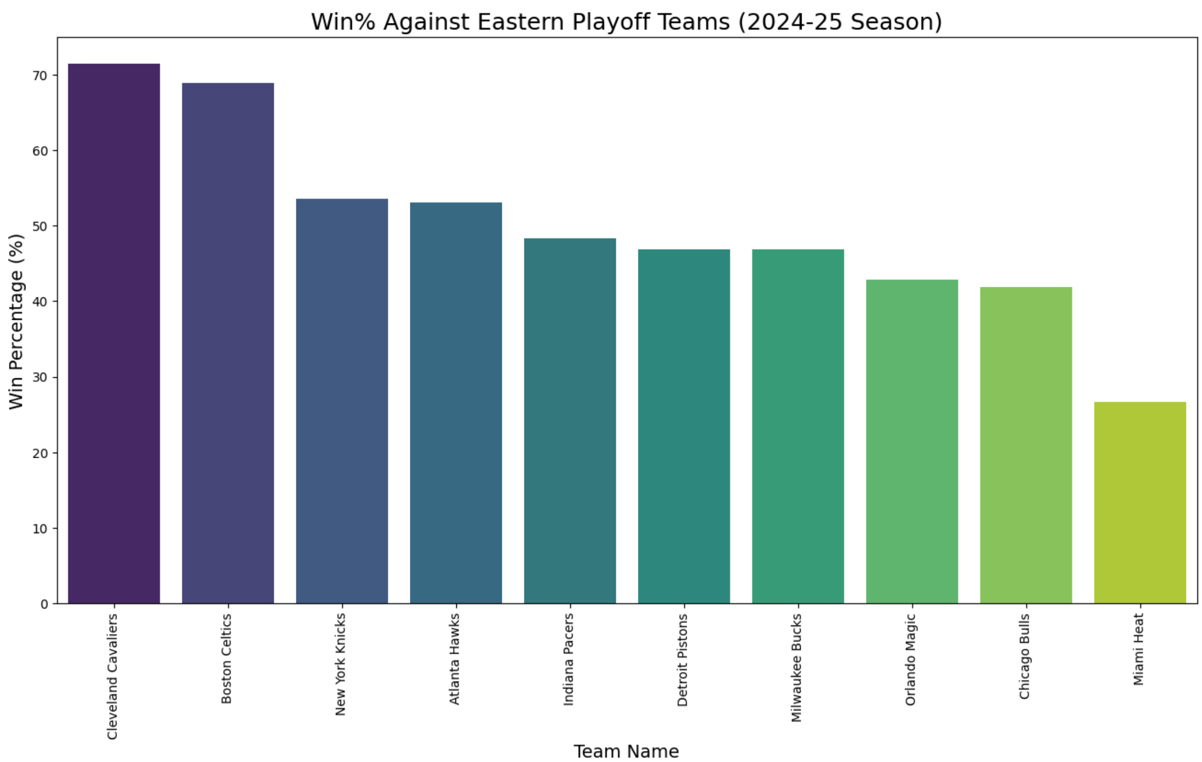
| | SEASON_ID | TEAM_ID | TEAM_ABBREVIATION | TEAM_NAME | GAME_ID | GAME_DATE | M/ |
|---|---|---|---|---|---|---|---|
| **26** | 22024 | 1610612749 | MIL | Milwaukee Bucks | 0022401083 | 2025-03-30 | |
| **27** | 22024 | 1610612737 | ATL | Atlanta Hawks | 0022401083 | 2025-03-30 | A' |
| **50** | 22024 | 1610612752 | NYK | New York Knicks | 0022401070 | 2025-03-28 | |
| **51** | 22024 | 1610612739 | CLE | Cleveland Cavaliers | 0022401067 | 2025-03-28 | |
| **55** | 22024 | 1610612749 | MIL | Milwaukee Bucks | 0022401070 | 2025-03-28 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **2233** | 22024 | 1610612754 | IND | Indiana Pacers | 0022400063 | 2024-10-23 | IN |
| **2241** | 22024 | 1610612753 | ORL | Orlando Magic | 0022400065 | 2024-10-23 | |
| **2246** | 22024 | 1610612765 | DET | Detroit Pistons | 0022400063 | 2024-10-23 | |
| **2249** | 22024 | 1610612738 | BOS | Boston Celtics | 0022400061 | 2024-10-22 | |
| **2250** | 22024 | 1610612752 | NYK | New York Knicks | 0022400061 | 2024-10-22 | |

301 rows × 29 columns

In [139]:
```
# Calculate total wins and games for all eastern playoff teams
eastern_wins = eastern_registered[eastern_registered['WL'] == 'W'].group
eastern_total_games = eastern_registered.groupby('TEAM_NAME').size()

# Calculate Win Percentage
eastern_win_percentage = (eastern_wins / eastern_total_games) * 100
eastern_win_percentage = eastern_win_percentage.fillna(0).sort_values(as

# Convert to DataFrame
eastern_win_df = eastern_win_percentage.reset_index()
eastern_win_df.columns = ['TEAM_NAME', 'Win% Against Eastern Playoff Tea
```

In [140]:
```
plt.figure(figsize=(16, 8))
sns.barplot(x='TEAM_NAME', y='Win% Against Eastern Playoff Teams', data=

plt.title('Win% Against Eastern Playoff Teams (2024-25 Season)', fontsiz
plt.xlabel('Team Name', fontsize=14)
plt.ylabel('Win Percentage (%)', fontsize=14)
plt.xticks(rotation=90)
plt.show()
```

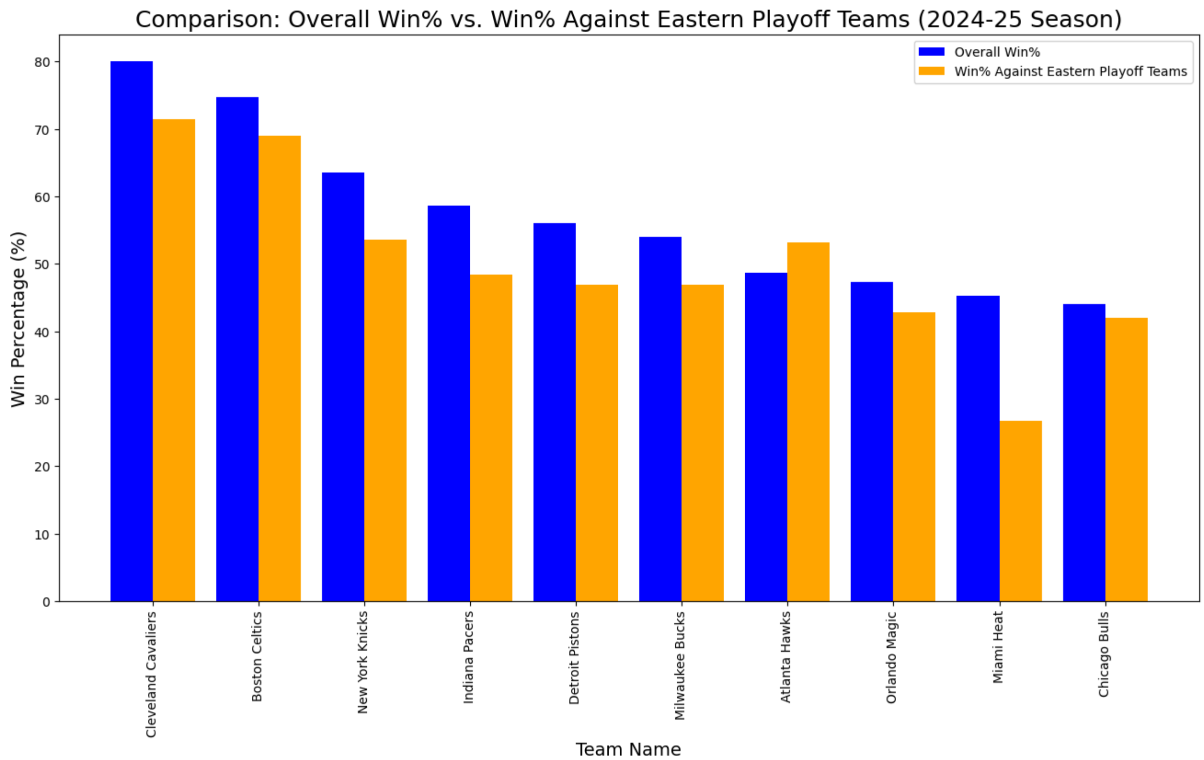
```
/var/folders/5n/r6b07v4s5cs3379jq_dtb8q00000gn/T/ipykernel_8135/1776064
726.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be rem
oved in v0.14.0. Assign the `x` variable to `hue` and set `legend=False
` for the same effect.

  sns.barplot(x='TEAM_NAME', y='Win% Against Eastern Playoff Teams', da
ta=eastern_win_df, palette='viridis')
```



In [141]:
```
# Calculate total wins and games for all eastern playoff teams
eastern_total_wins = df_schedule[(df_schedule['TEAM_NAME'].isin(eastern_
eastern_total_games = df_schedule[df_schedule['TEAM_NAME'].isin(eastern_

# Calculate Overall Win Percentage
eastern_overall_win_percentage = (eastern_total_wins / eastern_total_gam
eastern_overall_win_percentage = eastern_overall_win_percentage.fillna(0

# Convert to DataFrame
eastern_overall_win_df = eastern_overall_win_percentage.reset_index()
eastern_overall_win_df.columns = ['TEAM_NAME', 'Overall Win%']
```

In [142]:
```
# Merge the overall win percentage with the playoff-only win percentage
comparison_df2 = pd.merge(eastern_win_df, eastern_overall_win_df, on='TE

# Sort by Overall Win% for consistency
comparison_df2 = comparison_df2.sort_values(by='Overall Win%', ascending
```

In [143]:
```
plt.figure(figsize=(16, 8))
bar_width = 0.4
indices = range(len(comparison_df2))

# Plotting the two win percentages side by side
plt.bar(indices, comparison_df2['Overall Win%'], width=bar_width, label=
plt.bar([i + bar_width for i in indices], comparison_df2['Win% Against E

plt.title('Comparison: Overall Win% vs. Win% Against Eastern Playoff Tea
plt.xlabel('Team Name', fontsize=14)
plt.ylabel('Win Percentage (%)', fontsize=14)
plt.xticks([i + bar_width / 2 for i in indices], comparison_df2['TEAM_NA
plt.legend()
plt.show()
```




In [ ]: