In [24]:
```
import pandas as pd
import os

# Setup

# Define the directory where the processed (cleaned) data is stored
processed_data_dir = r"C:\Users\matth\OneDrive\Documents\data_science_

# Define the path to the combined team stats file
team_stats_file = os.path.join(processed_data_dir, "all_seasons_team_d

# Define the directory to save analysis outputs (figures, tables)
output_dir = r"C:\Users\matth\OneDrive\Documents\data_science_project\
figures_dir = os.path.join(output_dir, "figures")
tables_dir = os.path.join(output_dir, "tables")

# Create output directories if they don't exist
os.makedirs(figures_dir, exist_ok=True)
os.makedirs(tables_dir, exist_ok=True)

print("Output directories ensured exist:")
print(f"- Figures: {figures_dir}")
print(f"- Tables: {tables_dir}")
```

```
Output directories ensured exist:
- Figures: C:\Users\matth\OneDrive\Documents\data_science_project\pre
mier-league-home-advantage\output\figures
- Tables: C:\Users\matth\OneDrive\Documents\data_science_project\prem
ier-league-home-advantage\output\tables
```

In [25]:
```
# Load Data

# Load the combined team stats data
try:
    team_stats_df = pd.read_csv(team_stats_file, encoding='utf-8')
    print(f"\nSuccessfully loaded team stats data from: {team_stats_fi
except FileNotFoundError:
    print(f"Error: File not found at {team_stats_file}. Please ensure
    # Exit or handle error appropriately if the file is critical
    team_stats_df = None
except Exception as e:
    print(f"An error occurred loading the team stats data: {e}")
    team_stats_df = None
```

```
Successfully loaded team stats data from: C:\Users\matth\OneDrive\Doc
uments\data_science_project\premier-league-home-advantage\data\proces
sed_data\all_seasons_team_data.csv
```

In [26]:

```
# Analysis: Overall Home Advantage (League Average 2018-2024)

if team_stats_df is not None:
    print("\nCalculating overall league average home advantage (2018-2

    # Calculate overall average Home Points Per Game (PPG)
    overall_avg_home_ppg = team_stats_df['home_points_avg'].mean()

    # Calculate overall average Away Points Per Game (PPG)
    overall_avg_away_ppg = team_stats_df['away_points_avg'].mean()

    # Calculate the overall average PPG difference (Home - Away)
    overall_ppg_diff = overall_avg_home_ppg - overall_avg_away_ppg

    # Calculate overall average Home xG Difference per 90
    overall_avg_home_xg_diff_90 = team_stats_df['home_xg_diff_per90'].

    # Calculate overall average Away xG Difference per 90
    overall_avg_away_xg_diff_90 = team_stats_df['away_xg_diff_per90'].

    # Calculate the overall average xG Difference per 90 difference (H
    overall_xg_diff_90_diff = overall_avg_home_xg_diff_90 - overall_av

    # Print the results
    print("\n--- Overall League Average Home Advantage (2018-2024) ---
    print(f"Average Home PPG: {overall_avg_home_ppg:.2f}")
    print(f"Average Away PPG: {overall_avg_away_ppg:.2f}")
    print(f"Average Home Advantage (PPG Difference): {overall_ppg_diff
    print("-" * 60)
    print(f"Average Home xG Difference per 90: {overall_avg_home_xg_di
    print(f"Average Away xG Difference per 90: {overall_avg_away_xg_di
    print(f"Average Home Advantage (xG Diff per 90 Difference): {overa
    print("-" * 60)
else:
    print("\nSkipping analysis because team stats data failed to load.
```

```
Calculating overall league average home advantage (2018-2024)...

--- Overall League Average Home Advantage (2018-2024) ---
Average Home PPG: 1.56
Average Away PPG: 1.22
Average Home Advantage (PPG Difference): 0.34 points per game
------------------------------------------------------------
Average Home xG Difference per 90: 0.26
Average Away xG Difference per 90: -0.26
Average Home Advantage (xG Diff per 90 Difference): 0.53 xG per 90 mi
nutes
------------------------------------------------------------
```

In [4]:

```
# Analysis: Team-Specific Home Advantage (2018-2024)

if team_stats_df is not None:
    print("\nCalculating team-specific average home advantage (2018-20

    # Group by team and calculate the mean of the relevant metrics acr
    team_avg_stats = team_stats_df.groupby('team')[[
        'home_points_avg', 'away_points_avg',
        'home_xg_diff_per90', 'away_xg_diff_per90'
    ]].mean().reset_index() # reset_index turns the grouped 'team' bac

    # Calculate the PPG difference (Home Advantage in Points) for each
    team_avg_stats['ppg_diff'] = team_avg_stats['home_points_avg'] - t

    # Calculate the xG Difference per 90 difference (Home Advantage in
    team_avg_stats['xg_diff_90_diff'] = team_avg_stats['home_xg_diff_p

    print(" Team-specific averages and differences calculated.")
    # Display first few rows to check the result
    print(team_avg_stats.head())
else:
    print("\nSkipping calculation because team stats data failed to lo
```

```
Calculating team-specific average home advantage (2018-2024)...
 Team-specific averages and differences calculated.
          team  home_points_avg  away_points_avg  home_xg_diff_per90
\
0      Arsenal         2.121667         1.640000            0.710000
1  Aston Villa         1.580000         1.200000            0.120000
2  Bournemouth         1.305000         0.880000           -0.060000
3    Brentford         1.456667         1.070000            0.313333
4     Brighton         1.335000         1.113333            0.353333

   away_xg_diff_per90  ppg_diff  xg_diff_90_diff
0            0.158333  0.481667         0.551667
1           -0.388000  0.380000         0.508000
2           -0.582500  0.425000         0.522500
3           -0.190000  0.386667         0.503333
4           -0.265000  0.221667         0.618333
```

In [27]:

```
# Prepare Data for Top/Bottom 10 Visualisation (Based on PPG Diff)

if 'team_avg_stats' in locals() and team_avg_stats is not None:
     # Sort teams by PPG difference (highest home advantage first)
    team_avg_stats_sorted_ppg = team_avg_stats.sort_values(by='ppg_dif

    # Select the Top 10 teams
    top_10_ppg = team_avg_stats_sorted_ppg.head(10)

    # Select the Bottom 10 teams (lowest PPG difference, potentially n
    # Note: We take the tail and then sort it ascending for plotting c
    bottom_10_ppg = team_avg_stats_sorted_ppg.tail(10).sort_values(by=

    print(f"\nPrepared Top 10 and Bottom 10 teams based on PPG differe

    # Display the top/bottom tables
    print("\nTop 10 Teams (PPG Diff):")
    print(top_10_ppg[['team', 'ppg_diff']])
    print("\nBottom 10 Teams (PPG Diff):")
    print(bottom_10_ppg[['team', 'ppg_diff']])
else:
    print("\nSkipping visualisation preparation because team_avg_stats
```

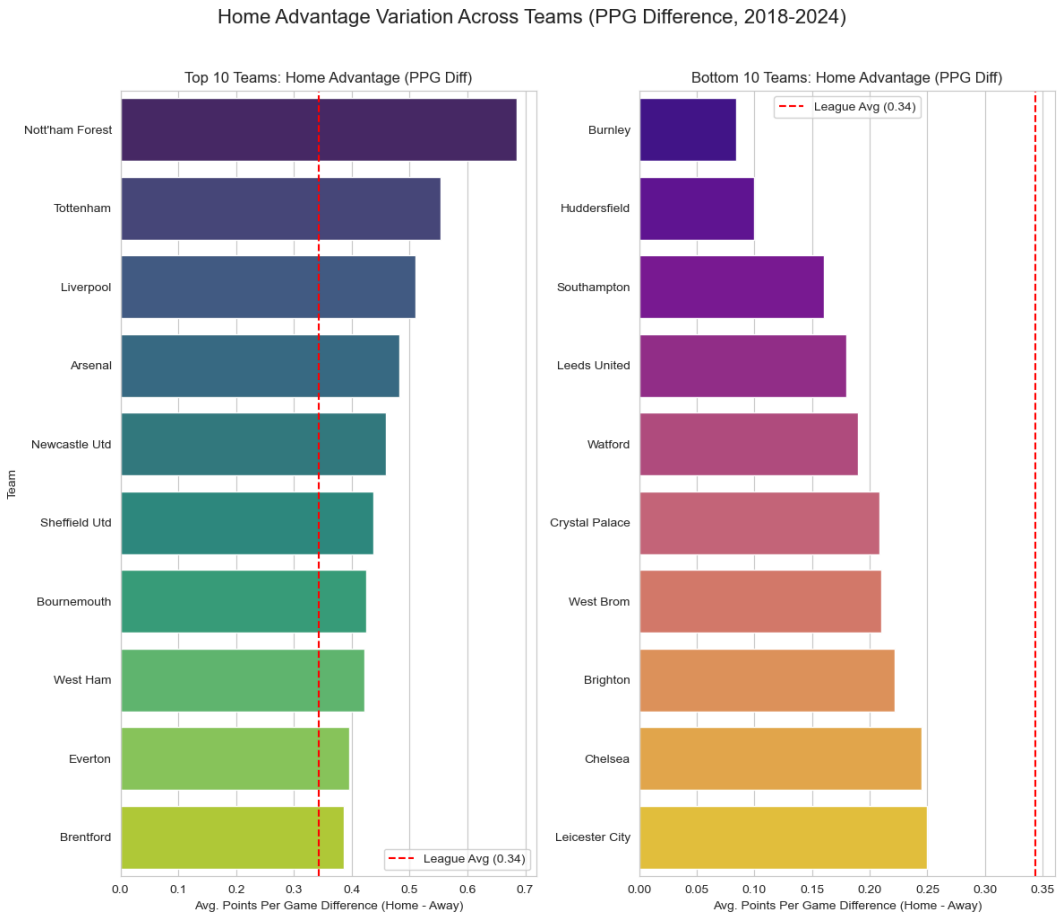
```
Prepared Top 10 and Bottom 10 teams based on PPG difference.

Top 10 Teams (PPG Diff):
               team  ppg_diff
20  Nott'ham Forest  0.685000
23        Tottenham  0.553333
14        Liverpool  0.510000
0           Arsenal  0.481667
18    Newcastle Utd  0.458333
21    Sheffield Utd  0.436667
2       Bournemouth  0.425000
26         West Ham  0.421667
9           Everton  0.395000
3         Brentford  0.386667

Bottom 10 Teams (PPG Diff):
              team  ppg_diff
5          Burnley  0.084000
11    Huddersfield  0.100000
22     Southampton  0.160000
12    Leeds United  0.180000
24         Watford  0.190000
8   Crystal Palace  0.208333
25       West Brom  0.210000
4         Brighton  0.221667
7          Chelsea  0.245000
13  Leicester City  0.250000
```

In [28]:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Visualise Top 10 / Bottom 10 Home Advantage (PPG Diff)

# Check if the dataframes for plotting exist from the previous cell
if 'top_10_ppg' in locals() and 'bottom_10_ppg' in locals() and 'overa
    print("\nGenerating visualization for Top/Bottom 10 teams...")

    # Set the style for the plots
    sns.set_style("whitegrid")
    plt.figure(figsize=(12, 10))

    # Subplot 1: Top 10 Teams
    plt.subplot(1, 2, 1) # 1 row, 2 columns, 1st subplot

    # Plot bars for Top 10
    sns.barplot(x='ppg_diff', y='team', data=top_10_ppg, palette='viri

    # Add titles and labels
    plt.title('Top 10 Teams: Home Advantage (PPG Diff)')
    plt.xlabel('Avg. Points Per Game Difference (Home - Away)')
    plt.ylabel('Team')

    # Add league average line and its legend
    plt.axvline(x=overall_ppg_diff, color='red', linestyle='--', label
    plt.legend()


    # Subplot 2: Bottom 10 Teams
    plt.subplot(1, 2, 2) # 1 row, 2 columns, 2nd subplot

    # Plot bars for Bottom 10
    sns.barplot(x='ppg_diff', y='team', data=bottom_10_ppg, palette='p

    # Add titles and labels
    plt.title('Bottom 10 Teams: Home Advantage (PPG Diff)')
    plt.xlabel('Avg. Points Per Game Difference (Home - Away)')
    plt.ylabel('') # Remove y-label for the second plot for cleaner lo

    # Add league average line and its legend
    plt.axvline(x=overall_ppg_diff, color='red', linestyle='--', label
    plt.legend()


    # Final Touches & Saving

    # Add overall title
    plt.suptitle('Home Advantage Variation Across Teams (PPG Differenc
    # Adjust layout to prevent overlap
    plt.tight_layout(rect=[0, 0, 1, 1]) # Adjust rect if suptitle over


    # Define the filename for the plot
    plot_filename = os.path.join(figures_dir, 'top_bottom_10_ppg_diff.

    # Save the plot
    try:
        plt.savefig(plot_filename, bbox_inches='tight', dpi=300) # Use
        print(f" Plot saved successfully to: {plot_filename}")
    except Exception as e:
```

```
        print(f" An error occurred while saving the plot: {e}")

    # Display the plot in the notebook
    plt.show()

else:
    print("\nSkipping visualisation because required dataframes (top_1
    print(" Ensure the previous cells calculating these have been run
```

```
Generating visualization for Top/Bottom 10 teams...
 Plot saved successfully to: C:\Users\matth\OneDrive\Documents\data_s
cience_project\premier-league-home-advantage\output\figures\top_botto
m_10_ppg_diff.png
```



In [29]:

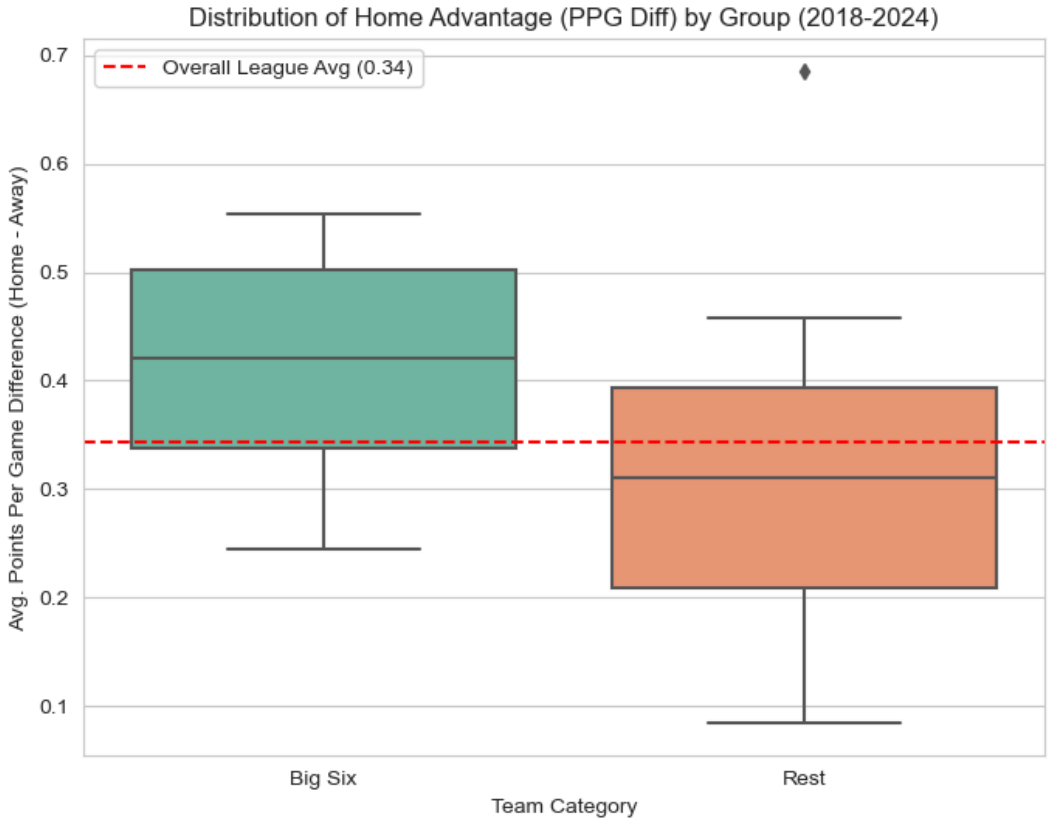
```
# Analysis: Big Six vs. Rest Comparison (Calculation)

# Check if the team_avg_stats DataFrame exists from Cell 2
if 'team_avg_stats' in locals() and team_avg_stats is not None:
    print("\nAnalyzing difference between 'Big Six' and other teams...

    # Define the list of "Big Six" teams
    big_six_teams = [
        "Manchester City",
        "Manchester Utd",
        "Liverpool",
        "Chelsea",
        "Arsenal",
        "Tottenham"
    ]
    print(f" Defined Big Six teams as: {big_six_teams}")

    # Create 'category' column if it doesn't exist
    if 'category' not in team_avg_stats.columns:
        team_avg_stats['category'] = team_avg_stats['team'].apply(
            lambda x: 'Big Six' if x in big_six_teams else 'Rest'
        )
        print(" Categorised teams into 'Big Six' and 'Rest'.")
    else:
        # If the column exists, just print a message
        print(" 'category' column already exists.")


    # Calculate the average home advantage metrics for each category
    group_avg_stats = team_avg_stats.groupby('category')[[
        'ppg_diff',
        'xg_diff_90_diff'
    ]].mean().reset_index()

    print("\n--- Average Home Advantage Metrics by Group (2018-2024) -
    print(group_avg_stats)
    print("-" * 60)

else:
    print("\nSkipping Big Six calculation because team_avg_stats DataF
    print(" Ensure the previous cells calculating team_avg_stats have
```

```
Analyzing difference between 'Big Six' and other teams...
 Defined Big Six teams as: ['Manchester City', 'Manchester Utd', 'Liv
erpool', 'Chelsea', 'Arsenal', 'Tottenham']
 'category' column already exists.

--- Average Home Advantage Metrics by Group (2018-2024) ---
  category  ppg_diff  xg_diff_90_diff
0  Big Six  0.413333         0.616389
1     Rest  0.309462         0.499523
------------------------------------------------------------
```

In [30]:

```
# Analysis: Big Six vs. Rest Comparison (Visualisation - Distribution

# Check if team_avg_stats DataFrame exists and has the 'category' colu
# Also check if overall_ppg_diff exists for the average line
if 'team_avg_stats' in locals() and team_avg_stats is not None and 'ca
    print("\nGenerating box plot for home advantage distribution (PPG

    # Set figure size
    plt.figure(figsize=(8, 6)) # Adjust size as needed

    # Create the box plot
    sns.boxplot(x='category', y='ppg_diff', data=team_avg_stats, palet

    # Add titles and labels
    plt.title('Distribution of Home Advantage (PPG Diff) by Group (201
    plt.xlabel('Team Category')
    plt.ylabel('Avg. Points Per Game Difference (Home - Away)')

    # Add overall league average line
    plt.axhline(y=overall_ppg_diff, color='red', linestyle='--', label
    plt.legend() # Show legend for the average line

    # Define filename and save
    # Using the original box plot filename
    plot_filename_box = os.path.join(figures_dir, 'big_six_rest_boxplo
    try:
        plt.savefig(plot_filename_box, bbox_inches='tight', dpi=300)
        print(f" Box plot saved successfully to: {plot_filename_box}")
    except Exception as e:
        print(f" An error occurred while saving the box plot: {e}")

    # Display the plot
    plt.show()

else:
    print("\nSkipping box plot generation because required data (team_
    print(" Ensure the previous cells have been run successfully.")
```

```
Generating box plot for home advantage distribution (PPG Diff)...
 Box plot saved successfully to: C:\Users\matth\OneDrive\Documents\da
ta_science_project\premier-league-home-advantage\output\figures\big_s
ix_rest_boxplot_ppg_diff.png
```


Distribution of Home Advantage (PPG Diff) by Group (2018-2024)
Overall League Avg (0.34)
Avg. Points Per Game Difference (Home - Away)
0.7
0.6
0.5
0.4
0.3
0.2
0.1
Big Six
Rest
Team Category

In [31]:

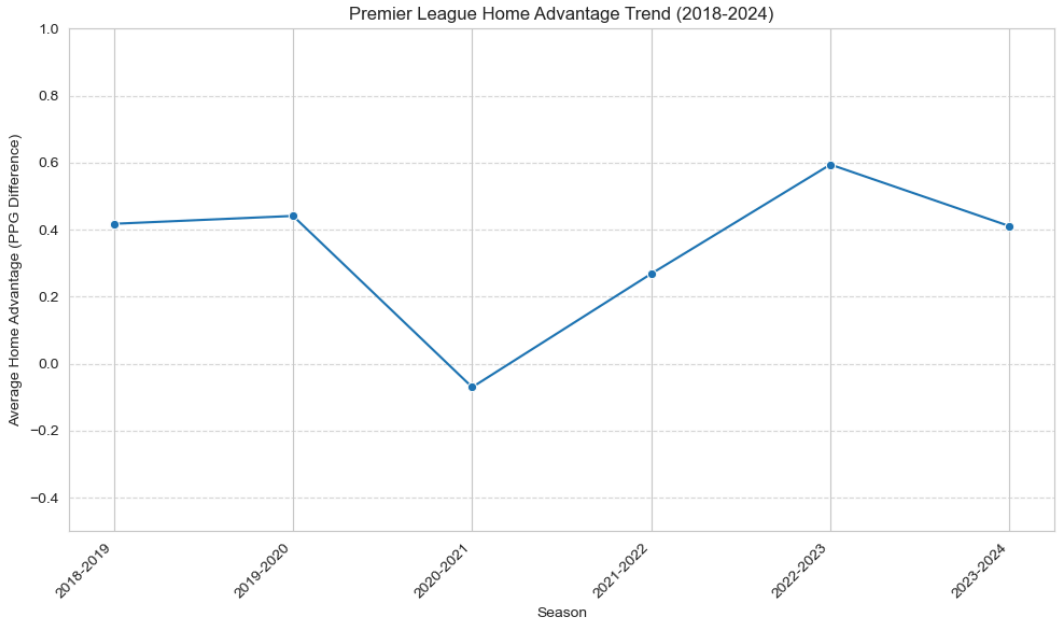
```
# Analysis: Home Advantage Trend Over Time (Calculation)

# Check if the team_stats_df DataFrame exists (loaded from all_seasons
if 'team_stats_df' in locals() and team_stats_df is not None:
    print("\nCalculating home advantage trend over seasons (2018-2024)

    # Calculate Average Home Advantage per Season
    # Group by season and calculate the mean of home/away PPG
    seasonal_avg_stats = team_stats_df.groupby('season')[[
        'home_points_avg', 'away_points_avg'
    ]].mean().reset_index()

    # Calculate the PPG difference (Home Advantage) for each season
    seasonal_avg_stats['seasonal_ppg_diff'] = seasonal_avg_stats['home

    print("\n--- Average Home Advantage (PPG Diff) per Season ---")
    # Display the calculated values for reference
    print(seasonal_avg_stats[['season', 'seasonal_ppg_diff']])
    print("-" * 60)

else:
    print("\nSkipping trend calculation because team_stats_df DataFram
    print(" Ensure the cell loading this data has been run successfull
```

```
Calculating home advantage trend over seasons (2018-2024)...

--- Average Home Advantage (PPG Diff) per Season ---
      season  seasonal_ppg_diff
0  2018-2019             0.4175
1  2019-2020             0.4410
2  2020-2021            -0.0705
3  2021-2022             0.2685
4  2022-2023             0.5945
5  2023-2024             0.4105
------------------------------------------------------------
```

In [32]:

```
# Analysis: Home Advantage Trend Over Time (Visualisation)

# Check if the seasonal_avg_stats DataFrame exists from the previous c
if 'seasonal_avg_stats' in locals() and seasonal_avg_stats is not None
    print("\nGenerating line plot for home advantage trend...")

    # Set figure size
    plt.figure(figsize=(10, 6)) # Adjust size as needed

    # Create the line plot
    sns.lineplot(x='season', y='seasonal_ppg_diff', data=seasonal_avg_

    # Add Titles and Labels
    plt.title('Premier League Home Advantage Trend (2018-2024)')
    plt.xlabel('Season')
    plt.ylabel('Average Home Advantage (PPG Difference)')
    plt.xticks(rotation=45, ha='right') # Rotate x-axis labels for bet
    plt.grid(axis='y', linestyle='--', alpha=0.7) # Add horizontal gri

    # Set Y-axis Limits
    # Set the limits for the y-axis as requested
    plt.ylim(-0.5, 1.0)
    print(" Set y-axis limits to (-0.5, 1.0).")

    # Define Filename and Save
    plot_filename_trend = os.path.join(figures_dir, 'home_advantage_tr
    try:
        plt.savefig(plot_filename_trend, bbox_inches='tight', dpi=300)
        print(f" Trend plot saved successfully to: {plot_filename_tren
    except Exception as e:
        print(f" An error occurred while saving the trend plot: {e}")

    # Display the plot
    plt.tight_layout() # Adjust layout after rotation and setting limi
    plt.show()

else:
    print("\nSkipping trend visualisation because seasonal_avg_stats D
    print(" Ensure the previous cell calculating this has been run suc
```

```
Generating line plot for home advantage trend...
 Set y-axis limits to (-0.5, 1.0).
 Trend plot saved successfully to: C:\Users\matth\OneDrive\Documents
\data_science_project\premier-league-home-advantage\output\figures\ho
me_advantage_trend_ppg.png
```


Premier League Home Advantage Trend (2018-2024)
Average Home Advantage (PPG Difference)
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
2018-2019
2019-2020
2020-2021
2021-2022
2022-2023
2023-2024
Season

In [ ]: