```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

html = urlopen('https://www.basketball-reference.com/leagues/NBA_2024_totals.html')
bs = BeautifulSoup(html.read(), 'html.parser')
print(bs)
```

```
<!DOCTYPE html>

<html class="no-js" data-root="/home/bbr/build" data-version="klecko-" lang="en">
<head>
<meta charset="utf-8"/>
<meta content="ie=edge" http-equiv="x-ua-compatible"/>
<meta content="width=device-width, initial-scale=1.0, maximum-scale=2.0" name="viewport">
<link href="https://cdn.ssref.net/req/202404081" rel="dns-prefetch"/>
<!-- InMobi Choice. Consent Manager Tag v3.0 (for TCF 2.2) -->
<script async="true" type="text/javascript">
(function() {
  var host = window.location.hostname;
  var element = document.createElement('script');
  var firstScript = document.getElementsByTagName('script')[0];
  var url = 'https://cmp.inmobi.com'
    .concat('/choice/', 'XwNYEpNeFfhfr', '/', host, '/choice.js?tag_version=V3');
  var uspTries = 0;
  var uspTriesLimit = 3;
  element.async = true;
  element.type = 'text/javascript';
  element.src = url;

  firstScript.parentNode.insertBefore(element, firstScript);

  function makeStub() {
    var TCF_LOCATOR_NAME = '__tcfapiLocator';
    var queue = [];
    var win = window;
    var cmpFrame;

    function addFrame() {
      var doc = win.document;
      var otherCMP = !!(win.frames[TCF_LOCATOR_NAME]);

      if (!otherCMP) {
        if (doc.body) {
          var iframe = doc.createElement('iframe');

          iframe.style.cssText = 'display:none';
          iframe.name = TCF_LOCATOR_NAME;
          doc.body.appendChild(iframe);
        } else {
          setTimeout(addFrame, 5);
        }
      }
      return !otherCMP;
    }

    function tcfAPIHandler() {
      var gdprApplies;
      var args = arguments;

      if (!args.length) {
        return queue;
      } else if (args[0] === 'setGdprApplies') {
        if (
```

```
# Find the table with the specified class
table = bs.find('table', class_='sortable stats_table')

# Check if the table is found
if table:
    rows = table.find_all('tr')
    header_columns = [th.text.strip() for th in rows[0].find_all('th')]

    # Extract the data from the remaining rows
    data = []
    for row in rows[1:]:
        if len(row.find_all(['td', 'th'])) == len(header_columns):
            row_data = [td.text.strip() for td in row.find_all(['td', 'th'])]
            data.append(row_data)

    # Convert header_columns and data to DataFrame
    df = pd.DataFrame(data, columns=header_columns)

    # Print the DataFrame
    print("DataFrame:")
    print(df)
else:
    print("Table not found.")
```

```
DataFrame:
       Rk            Player   Pos Age   Tm   G  GS    MP   FG   FGA  ...   FT%  \
0       1  Precious Achiuwa  PF-C  24  TOT  74  18  1624  235   469  ...  .616
1       1  Precious Achiuwa     C  24  TOR  25   0   437   78   170  ...  .571
2       1  Precious Achiuwa    PF  24  NYK  49  18  1187  157   299  ...  .643
3       2       Bam Adebayo     C  26  MIA  71  71  2416  530  1017  ...  .755
4       3      Ochai Agbaji    SG  23  TOT  78  28  1641  178   433  ...  .661
..    ...               ...   ...  ..  ...  ..  ..   ...  ...   ...  ...   ...
758   568    Thaddeus Young    PF  35  PHO  10   0    89   11    21  ...  .333
759   569        Trae Young    PG  25  ATL  54  54  1942  433  1008  ...  .855
760   570    Omer Yurtseven     C  25  UTA  48  12   545   99   184  ...  .679
761   571       Cody Zeller     C  31  NOP  43   0   320   26    62  ...  .605
762   572       Ivica Zubac     C  26  LAC  68  68  1794  337   519  ...  .723

     ORB  DRB  TRB  AST STL BLK  TOV   PF   PTS
0    191  296  487   97  46  68   83  143   565
1     50   86  136   44  16  12   29   40   193
2    141  210  351   53  30  56   54  103   372
3    159  578  737  278  81  66  162  159  1367
4     74  142  216   83  47  44   64  117   455
..   ...  ...  ...  ...  ..  ..  ...  ...   ...
758   17   11   28    7   5   2    4   11    23
759   23  126  149  583  72  11  235  109  1389
760   72  136  208   29   8  18   37   52   222
761   48   64  112   39   9   5   16   45    76
762  196  430  626   93  22  83   79  180   794

[763 rows x 30 columns]
```

```
print(df.columns)
```

```
Index(['Rk', 'Player', 'Pos', 'Age', 'Tm', 'G', 'GS', 'MP', 'FG', 'FGA', 'FG%',
       '3P', '3PA', '3P%', '2P', '2PA', '2P%', 'eFG%', 'FT', 'FTA', 'FT%',
       'ORB', 'DRB', 'TRB', 'AST', 'STL', 'BLK', 'TOV', 'PF', 'PTS'],
      dtype='object')
```

```
selected_columns = ["Player", "G", "PTS","TRB", "AST", "STL","TOV"]
df_filtered = df[selected_columns]
print(df_filtered)
```

```
                Player   G   PTS  TRB  AST STL  TOV
0    Precious Achiuwa  74   565  487   97  46   83
1    Precious Achiuwa  25   193  136   44  16   29
2    Precious Achiuwa  49   372  351   53  30   54
3         Bam Adebayo  71  1367  737  278  81  162
4        Ochai Agbaji  78   455  216   83  47   64
..                ...  ..   ...  ...  ...  ..  ...
758    Thaddeus Young  10    23   28    7   5    4
759        Trae Young  54  1389  149  583  72  235
760    Omer Yurtseven  48   222  208   29   8   37
761       Cody Zeller  43    76  112   39   9   16
762       Ivica Zubac  68   794  626   93  22   79

[763 rows x 7 columns]
```

```
column_names_to_sum = ['G', 'PTS', 'TRB', 'AST', 'STL', 'TOV']

# Convert the selected columns to numeric values
df_filtered_numeric = df_filtered[column_names_to_sum].apply(pd.to_numeric, errors='coerce')

# Add the 'Player' column back to the DataFrame
df_filtered_numeric['Player'] = df_filtered['Player']

# Drop duplicate rows based on 'Player' column
df_filtered_numeric = df_filtered_numeric.drop_duplicates(subset=['Player'])

# Ensure 'Player' column is of string type
df_filtered_numeric['Player'] = df_filtered_numeric['Player'].astype(str)

# Group by 'Player' and sum the statistics for each player
player_stats_sum = df_filtered_numeric.groupby('Player', as_index=False).sum()

# Print the summed statistics for each player
print(player_stats_sum)
```

```
                Player     G     PTS    TRB    AST   STL    TOV
0         A.J. Green  56.0   252.0   64.0   30.0   9.0   12.0
1        A.J. Lawson  42.0   136.0   50.0   20.0  10.0   14.0
2         AJ Griffin  20.0    48.0   18.0    5.0   1.0    8.0
3       Aaron Gordon  73.0  1013.0  471.0  259.0  56.0  105.0
4      Aaron Holiday  78.0   514.0  123.0  140.0  42.0   53.0
..               ...   ...     ...    ...    ...   ...    ...
568      Zach LaVine  25.0   487.0  129.0   98.0  21.0   52.0
569   Zavier Simpson   7.0    42.0   20.0   25.0   7.0   10.0
570       Zeke Nnaji  58.0   186.0  126.0   32.0  15.0   27.0
571  Ziaire Williams  51.0   420.0  180.0   75.0  36.0   66.0
572  Zion Williamson  70.0  1601.0  406.0  352.0  77.0  193.0

[573 rows x 7 columns]
```

```
column_names_to_divide = ['PTS', 'TRB', 'AST', 'STL', 'TOV', 'G']

# Convert the selected columns to numeric values
player_stats_sum[column_names_to_divide] = player_stats_sum[column_names_to_divide].apply(pd.to_numeric, errors='coerce')

# Create new columns by dividing each Games column
player_stats_sum['Points per Game'] = player_stats_sum['PTS'] / player_stats_sum['G']
player_stats_sum['Rebounds per Game'] = player_stats_sum['TRB'] / player_stats_sum['G']
player_stats_sum['Assists per Game'] = player_stats_sum['AST'] / player_stats_sum['G']
player_stats_sum['Steals per Game'] = player_stats_sum['STL'] / player_stats_sum['G']
player_stats_sum['Turnovers per Game'] = player_stats_sum['TOV'] / player_stats_sum['G']

# Drop the original columns
player_stats_sum = player_stats_sum.drop(['PTS', 'TRB', 'AST', 'STL', 'TOV'], axis=1)

# Print the updated DataFrame
print(player_stats_sum)
```

```
                Player     G  Points per Game  Rebounds per Game  \
0         A.J. Green  56.0         4.500000           1.142857
1        A.J. Lawson  42.0         3.238095           1.190476
2         AJ Griffin  20.0         2.400000           0.900000
3       Aaron Gordon  73.0        13.876712           6.452055
4      Aaron Holiday  78.0         6.589744           1.576923
..               ...   ...              ...                ...
568      Zach LaVine  25.0        19.480000           5.160000
569   Zavier Simpson   7.0         6.000000           2.857143
570       Zeke Nnaji  58.0         3.206897           2.172414
571  Ziaire Williams  51.0         8.235294           3.529412
572  Zion Williamson  70.0        22.871429           5.800000

     Assists per Game  Steals per Game  Turnovers per Game
0            0.535714         0.160714            0.214286
1            0.476190         0.238095            0.333333
2            0.250000         0.050000            0.400000
3            3.547945         0.767123            1.438356
4            1.794872         0.538462            0.679487
..                ...              ...                 ...
568          3.920000         0.840000            2.080000
569          3.571429         1.000000            1.428571
570          0.551724         0.258621            0.465517
571          1.470588         0.705882            1.294118
```

```
572          5.028571         1.100000            2.757143

[573 rows x 7 columns]
```

```
# Create new columns for each average
player_stats_sum['Average Points per Game'] = player_stats_sum['Points per Game'].mean()
player_stats_sum['Average Rebounds per Game'] = player_stats_sum['Rebounds per Game'].mean()
player_stats_sum['Average Assists per Game'] = player_stats_sum['Assists per Game'].mean()
player_stats_sum['Average Steals per Game'] = player_stats_sum['Steals per Game'].mean()
player_stats_sum['Average Turnovers per Game'] = player_stats_sum['Turnovers per Game'].mean()

# Print the updated DataFrame
print(player_stats_sum)
```

```
                Player     G  Points per Game  Rebounds per Game  \
0         A.J. Green  56.0         4.500000           1.142857
1        A.J. Lawson  42.0         3.238095           1.190476
2         AJ Griffin  20.0         2.400000           0.900000
3       Aaron Gordon  73.0        13.876712           6.452055
4      Aaron Holiday  78.0         6.589744           1.576923
..               ...   ...              ...                ...
568      Zach LaVine  25.0        19.480000           5.160000
569   Zavier Simpson   7.0         6.000000           2.857143
570       Zeke Nnaji  58.0         3.206897           2.172414
571  Ziaire Williams  51.0         8.235294           3.529412
572  Zion Williamson  70.0        22.871429           5.800000

     Assists per Game  Steals per Game  Turnovers per Game  \
0            0.535714         0.160714            0.214286
1            0.476190         0.238095            0.333333
2            0.250000         0.050000            0.400000
3            3.547945         0.767123            1.438356
4            1.794872         0.538462            0.679487
..                ...              ...                 ...
568          3.920000         0.840000            2.080000
569          3.571429         1.000000            1.428571
570          0.551724         0.258621            0.465517
571          1.470588         0.705882            1.294118
572          5.028571         1.100000            2.757143

     Average Points per Game  Average Rebounds per Game  \
0                     8.4223                   3.374702
1                     8.4223                   3.374702
2                     8.4223                   3.374702
3                     8.4223                   3.374702
4                     8.4223                   3.374702
..                       ...                        ...
568                   8.4223                   3.374702
569                   8.4223                   3.374702
570                   8.4223                   3.374702
571                   8.4223                   3.374702
572                   8.4223                   3.374702

     Average Assists per Game  Average Steals per Game  \
0                    2.001295                  0.59035
1                    2.001295                  0.59035
2                    2.001295                  0.59035
3                    2.001295                  0.59035
4                    2.001295                  0.59035
..                        ...                      ...
568                  2.001295                  0.59035
569                  2.001295                  0.59035
570                  2.001295                  0.59035
571                  2.001295                  0.59035
572                  2.001295                  0.59035

     Average Turnovers per Game
0                      0.981566
1                      0.981566
2                      0.981566
3                      0.981566
4                      0.981566
```

```
print(player_stats_sum.columns)
```

```
Index(['Player', 'G', 'Points per Game', 'Rebounds per Game',
       'Assists per Game', 'Steals per Game', 'Turnovers per Game',
       'Average Points per Game', 'Average Rebounds per Game',
       'Average Assists per Game', 'Average Steals per Game',
       'Average Turnovers per Game'],
      dtype='object')
```

```
# Define the list of rookies

rookies = [
    "Victor Wembanyama",
    "Brandon Miller",
    "Scoot Henderson",
    "Amen Thompson",
    "Ausar Thompson",
    "Anthony Black",
    "Bilal Coulibaly",
    "Jarace Walker",
    "Taylor Hendricks",
    "Cason Wallace",
    "Jett Howard",
    "Dereck Lively II",
    "Gradey Dick",
    "Jordan Hawkins",
    "Kobe Bufkin",
    "Keyonte George",
    "Jalen Hood-Schifino",
    "Jaime Jaquez Jr.",
    "Brandin Podziemski",
    "Cam Whitmore",
    "Noah Clowney",
    "Dariq Whitehead",
    "Kris Murray",
    "Olivier-Maxence Prosper",
    "Marcus Sasser",
    "Ben Sheppard",
    "Nick Smith Jr.",
    "Brice Sensabaugh",
    "Julian Strawther",
    "Kobe Brown"
]

# Filter the DataFrame to only include the players in the rookies list
rookies_df = player_stats_sum[player_stats_sum['Player'].isin(rookies)]
print(rookies_df)
```

```
                        Player     G  Points per Game  Rebounds per Game  \
19             Amen Thompson  62.0         9.548387           6.596774
27             Anthony Black  69.0         4.579710           2.014493
33            Ausar Thompson  63.0         8.825397           6.380952
37              Ben Sheppard  57.0         4.421053           1.561404
40           Bilal Coulibaly  63.0         8.444444           4.063492
50        Brandin Podziemski  74.0         9.216216           5.770270
54             Brandon Miller  74.0        17.283784           4.256757
57          Brice Sensabaugh  32.0         7.531250           3.187500
70              Cam Whitmore  47.0        12.319149           3.829787
74             Cason Wallace  82.0         6.841463           2.280488
117          Dariq Whitehead   2.0         1.500000           2.000000
137         Dereck Lively II  55.0         8.781818           6.872727
187              Gradey Dick  60.0         8.500000           2.200000
230         Jaime Jaquez Jr.  75.0        11.853333           3.800000
237      Jalen Hood-Schifino  21.0         1.619048           0.619048
253            Jarace Walker  33.0         3.636364           1.909091
278              Jett Howard  18.0         1.611111           0.388889
296           Jordan Hawkins  67.0         7.820896           2.208955
315         Julian Strawther  50.0         4.540000           1.220000
345           Keyonte George  75.0        12.986667           2.813333
350              Kobe Brown  44.0         2.022727           1.409091
351             Kobe Bufkin  17.0         4.764706           1.941176
354             Kris Murray  62.0         6.064516           3.612903
388            Marcus Sasser  71.0         8.253521           1.760563
431           Nick Smith Jr.  51.0         5.921569           1.411765
437             Noah Clowney  23.0         5.782609           3.521739
443  Olivier-Maxence Prosper  40.0         3.025000           1.975000
496          Scoot Henderson  62.0        14.000000           3.129032
518         Taylor Hendricks  40.0         7.300000           4.625000
556        Victor Wembanyama  71.0        21.436620          10.633803

     Assists per Game  Steals per Game  Turnovers per Game  \
19           2.629032         1.258065            1.451613
27           1.318841         0.507246            0.811594
33           1.904762         1.079365            1.333333
37           0.929825         0.578947            0.263158
40           1.746032         0.904762            1.380952
50           3.689189         0.824324            1.189189
54           2.364865         0.891892            1.783784
```

```
57          1.718750        0.406250          1.468750
70          0.702128        0.638298          0.978723
74          1.463415        0.926829          0.548780
117         1.500000        0.000000          0.000000
137         1.090909        0.654545          0.909091
187         1.133333        0.566667          0.833333
230         2.600000        1.026667          1.466667
237         0.380952        0.142857          0.428571
253         1.212121        0.454545          0.515152
278         0.333333        0.111111          0.166667
296         1.044776        0.283582          0.597015
315         0.940000        0.340000          0.460000
345         4.426667        0.480000          2.506667
350         0.568182        0.272727          0.204545
351         1.588235        0.411765          0.588235
354         1.290323        0.854839          0.887097
388         3.323944        0.619718          1.267606
431         1.156863        0.196078          0.764706
```

```
# Saving DataFrame to a CSV file
rookies_df.to_csv('final_project_405.csv', index=False)

print("DataFrame saved as CSV successfully!")
```

```
DataFrame saved as CSV successfully!
```

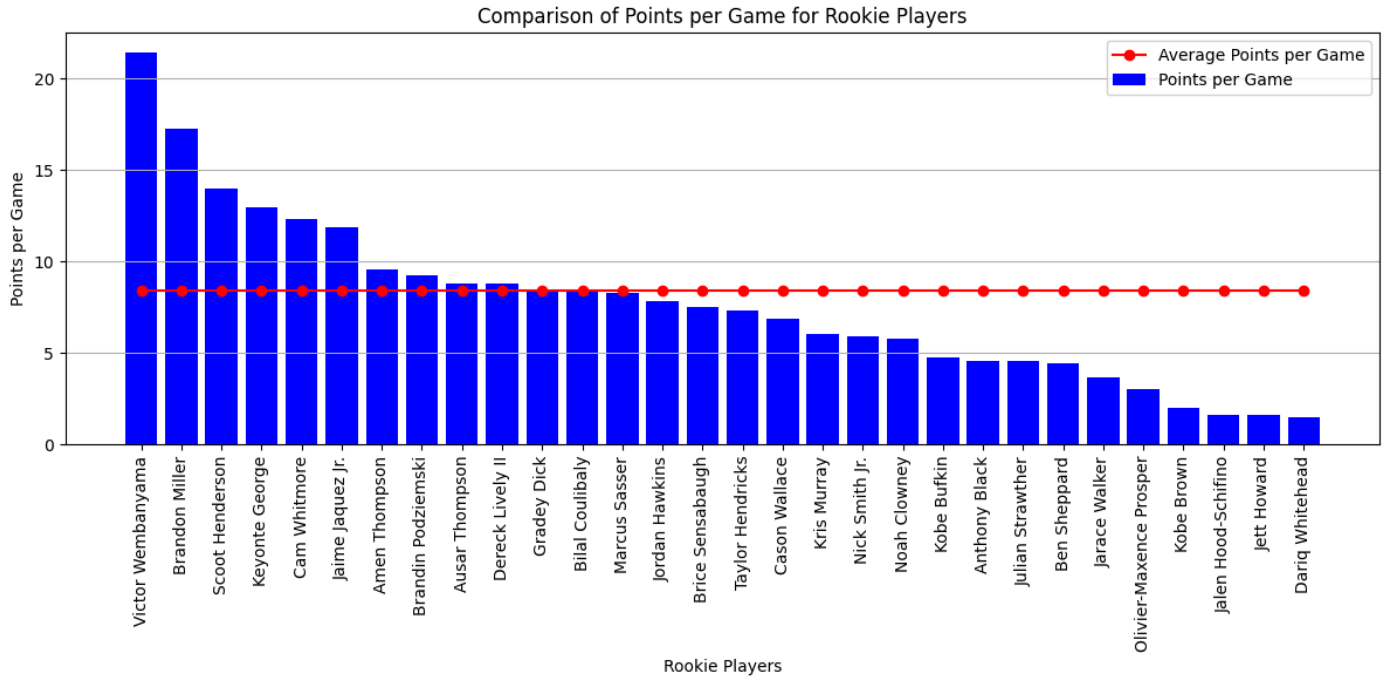
```
# Sort the DataFrame by points per game in descending order
rookies_df_sorted = rookies_df.sort_values(by='Points per Game', ascending=False)

# Extract relevant columns from the sorted DataFrame
players = rookies_df_sorted['Player']
points_per_game = rookies_df_sorted['Points per Game']
average_points_per_game = rookies_df_sorted['Average Points per Game']

# Plotting the data
plt.figure(figsize=(12, 6))
plt.bar(players, points_per_game, label='Points per Game', color='blue')
plt.plot(players, average_points_per_game, label='Average Points per Game', color='red', marker='o')

# Adding labels and title
plt.xlabel('Rookie Players')
plt.ylabel('Points per Game')
plt.title('Comparison of Points per Game for Rookie Players')
plt.xticks(rotation=90)
plt.legend()
plt.grid(axis='y')

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

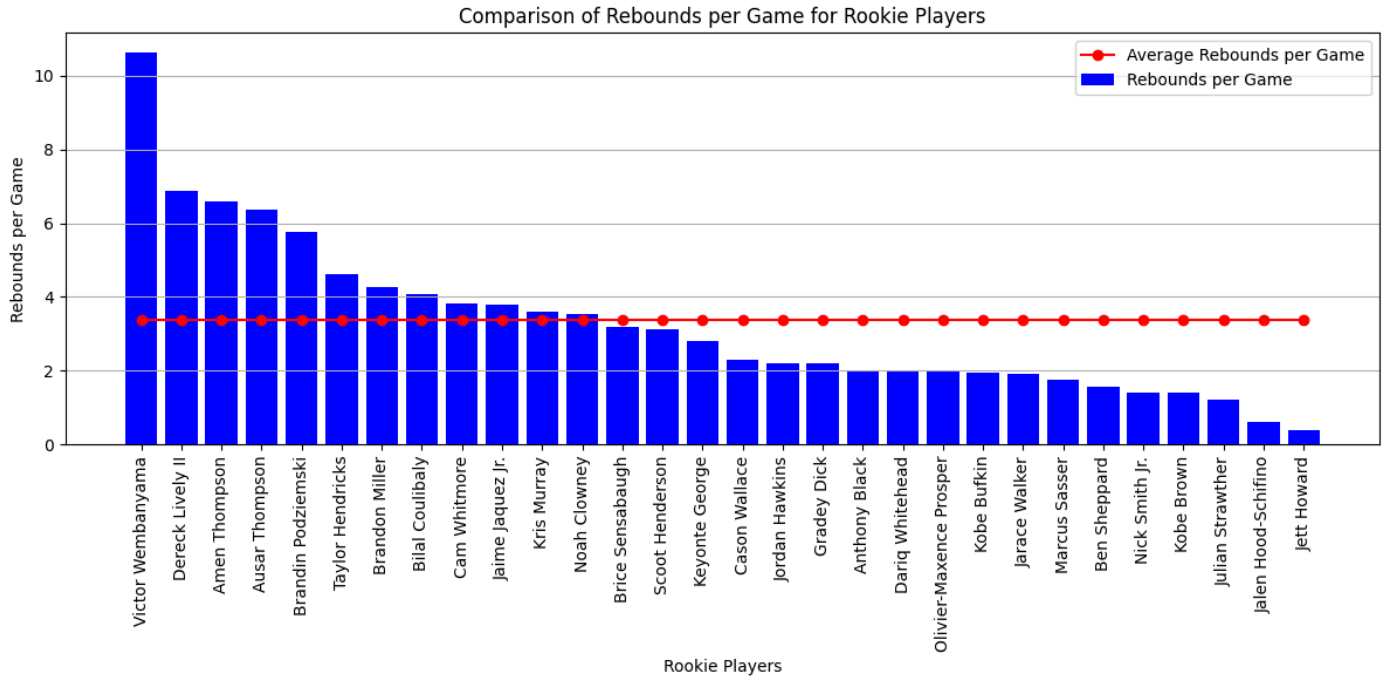



```
# Sort the DataFrame by rebounds per game in descending order
rookies_df_sorted = rookies_df.sort_values(by='Rebounds per Game', ascending=False)

# Extract relevant columns from the sorted DataFrame
players = rookies_df_sorted['Player']
rebounds_per_game = rookies_df_sorted['Rebounds per Game']
average_rebounds_per_game = rookies_df_sorted['Average Rebounds per Game']

# Plotting the data
plt.figure(figsize=(12, 6))
plt.bar(players, rebounds_per_game, label='Rebounds per Game', color='blue')
plt.plot(players, average_rebounds_per_game, label='Average Rebounds per Game', color='red', marker='o')

# Adding labels and title
plt.xlabel('Rookie Players')
plt.ylabel('Rebounds per Game')
plt.title('Comparison of Rebounds per Game for Rookie Players')
plt.xticks(rotation=90)
plt.legend()
plt.grid(axis='y')
# Show the plot
plt.tight_layout()
plt.show()
```

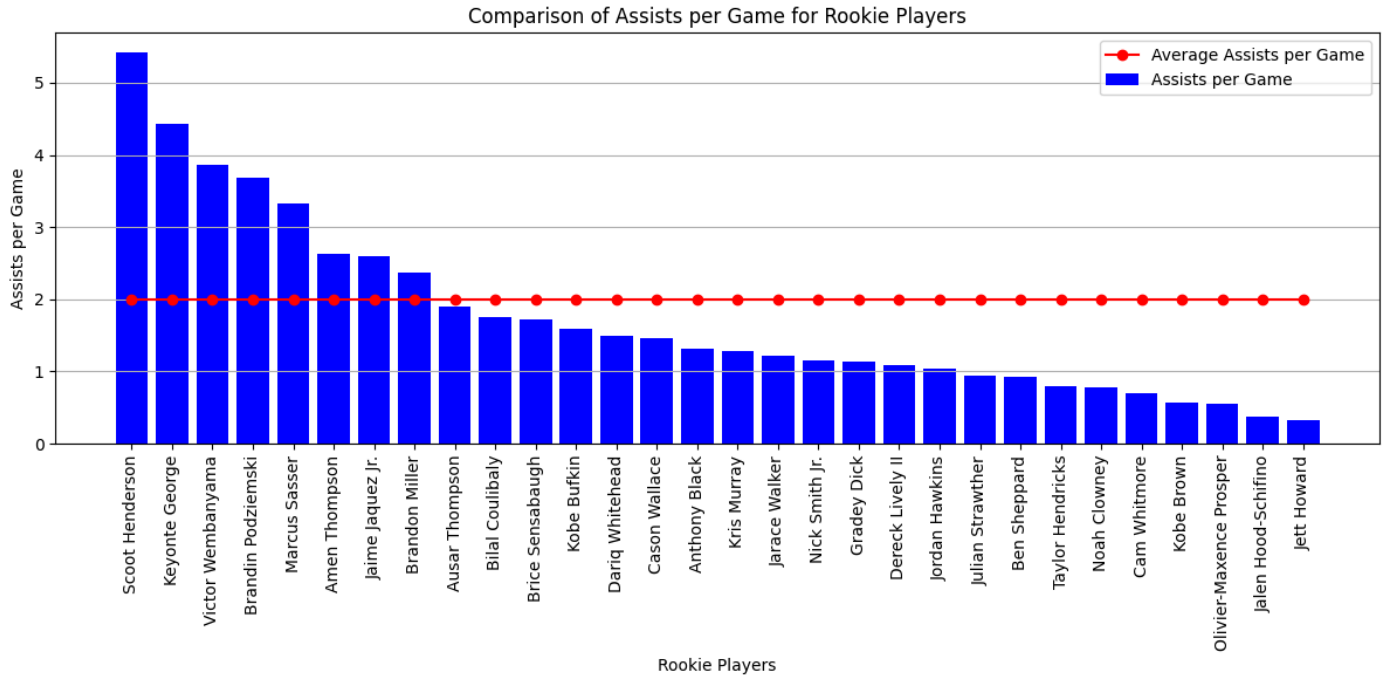



```
# Sort the DataFrame by rebounds per game in descending order
rookies_df_sorted = rookies_df.sort_values(by='Assists per Game', ascending=False)

# Extract relevant columns from the sorted DataFrame
players = rookies_df_sorted['Player']
rebounds_per_game = rookies_df_sorted['Assists per Game']
average_rebounds_per_game = rookies_df_sorted['Average Assists per Game']

# Plotting the data
plt.figure(figsize=(12, 6))
plt.bar(players, rebounds_per_game, label='Assists per Game', color='blue')
plt.plot(players, average_rebounds_per_game, label='Average Assists per Game', color='red', marker='o')

# Adding labels and title
plt.xlabel('Rookie Players')
plt.ylabel('Assists per Game')
plt.title('Comparison of Assists per Game for Rookie Players')
plt.xticks(rotation=90)
plt.legend()
plt.grid(axis='y')
# Show the plot
plt.tight_layout()
plt.show()
```

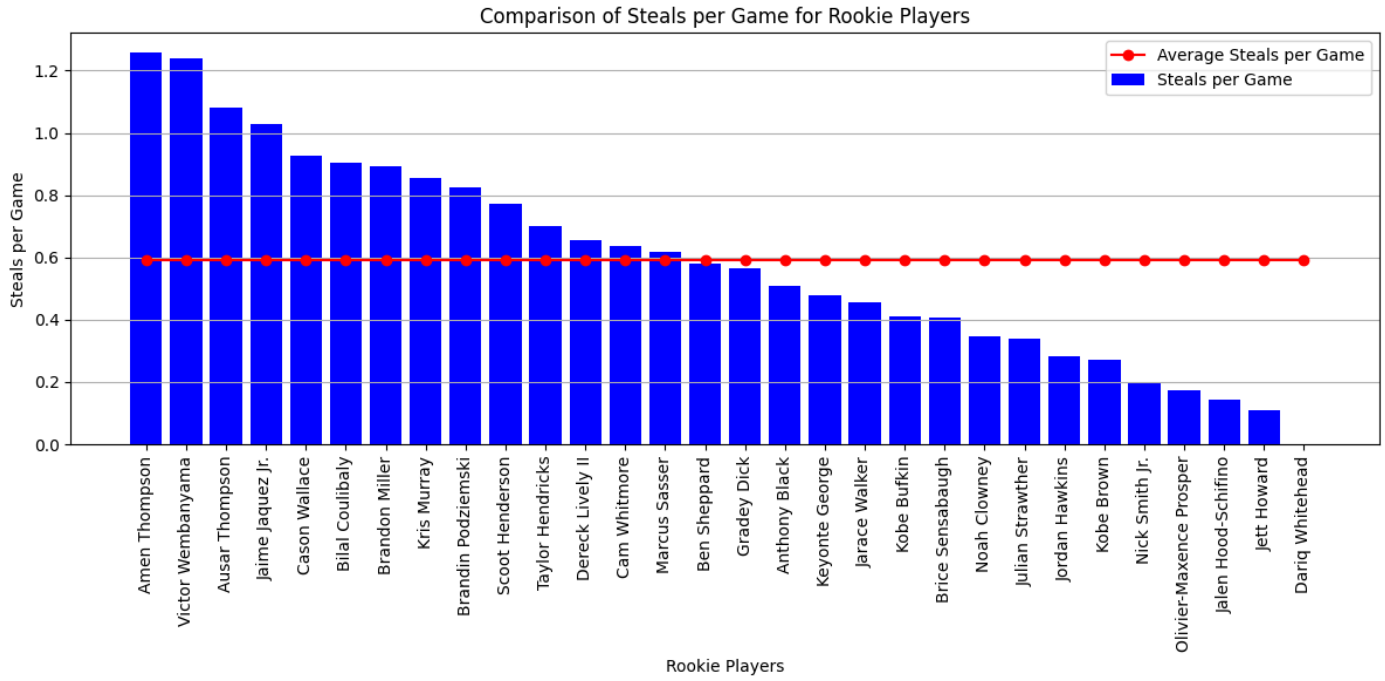



```
# Sort the DataFrame by rebounds per game in descending order
rookies_df_sorted = rookies_df.sort_values(by='Steals per Game', ascending=False)

# Extract relevant columns from the sorted DataFrame
players = rookies_df_sorted['Player']
rebounds_per_game = rookies_df_sorted['Steals per Game']
average_rebounds_per_game = rookies_df_sorted['Average Steals per Game']

# Plotting the data
plt.figure(figsize=(12, 6))
plt.bar(players, rebounds_per_game, label='Steals per Game', color='blue')
plt.plot(players, average_rebounds_per_game, label='Average Steals per Game', color='red', marker='o')

# Adding labels and title
plt.xlabel('Rookie Players')
plt.ylabel('Steals per Game')
plt.title('Comparison of Steals per Game for Rookie Players')
plt.xticks(rotation=90)
plt.legend()
plt.grid(axis='y')
# Show the plot
plt.tight_layout()
plt.show()
```



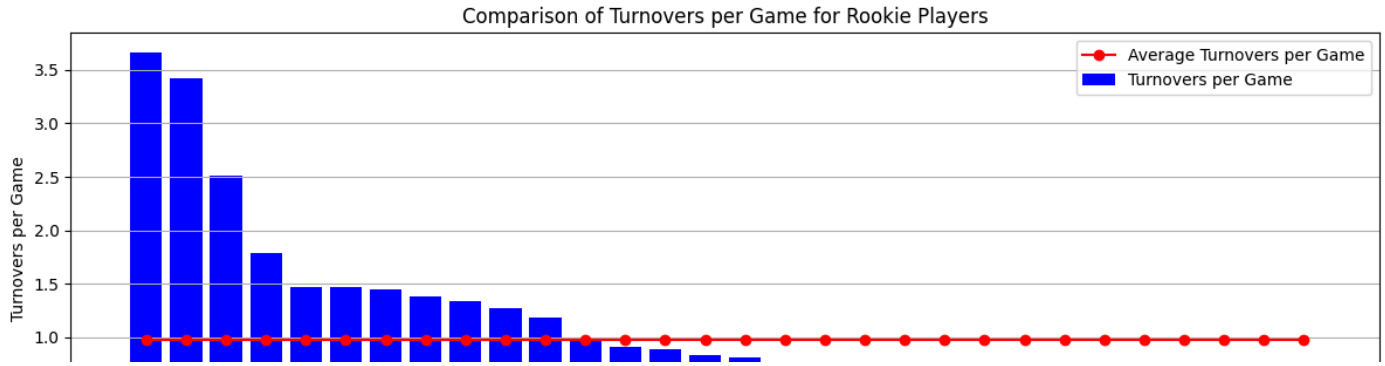

```
# Sort the DataFrame by rebounds per game in descending order
rookies_df_sorted = rookies_df.sort_values(by='Turnovers per Game', ascending=False)

# Extract relevant columns from the sorted DataFrame
players = rookies_df_sorted['Player']
rebounds_per_game = rookies_df_sorted['Turnovers per Game']
average_rebounds_per_game = rookies_df_sorted['Average Turnovers per Game']

# Plotting the data
plt.figure(figsize=(12, 6))
plt.bar(players, rebounds_per_game, label='Turnovers per Game', color='blue')
plt.plot(players, average_rebounds_per_game, label='Average Turnovers per Game', color='red', marker='o')

# Adding labels and title
plt.xlabel('Rookie Players')
plt.ylabel('Turnovers per Game')
plt.title('Comparison of Turnovers per Game for Rookie Players')
plt.xticks(rotation=90)
plt.legend()
plt.grid(axis='y')


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




```
# Create a new column 'Above Average' which is 1 if the player's points per game is greater than the league average, and 0 otherwise
rookies_df_sorted['Above Average'] = (rookies_df_sorted['Points per Game'] - rookies_df_sorted['Points per Game'].mean()).astype(int) + (roc
# Print the new DataFrame
rookies_df_sorted.head()
```