worldcup

May 1, 2023

1 2022 Qatar World Cup Analysis

The World Cup is one of the, if not the largest, sporting event in the world. The World Cup is a tournament consisting of some of the best qualifying nations in international football. Countries are divided into groups, where the top two teams from each group move on to the 'knockout rounds', effectively dwindling down the competition until one team is left standing.The World Cup is a global event that transcends borders, cultures, and languages, captivating audiences with the sheer talent, passion, and dedication of the world's top footballers. The World Cup also serves as a platform for countries to showcase their national identity and pride, and to unite their citizens around a common goal. Whether you are a casual fan or a die-hard supporter, the World Cup is an unforgettable experience that captures the imagination and inspires a sense of unity that extends far beyond the pitch. And for the players, it is considered the highest honor in the sport; one that demands a lot of passion and excellence in order to persevere and bring home glory to their country.

The 2022 World Cup in Qatar has recently concluded with an Argentinian extra-time victory in the final over France; a dramatic victory that perfectly captures the essence of the tournament. The dataset that we will be utilizing in our project focuses on each individual match from the tournament, and contains a large amount of data about the match itself, specifically for each team; possession, shots attempted, shots on goal, total passes, etc. We are hoping to showcase some trends, and insights that can summarize and visualize the tournament effectively through data.

Our goal is to provide some background on the tournament; the teams, the players, the matches, and the results. We will be using the dataset to answer some questions that we have about the tournament, and to provide some insights that can help us understand the tournament better.

Outside our background analysis, some questions we hope to answer are:

- Who were the most standout players in the tournament?
    - Who was the most efficient player throughout the tournament? (we can calculate some statistics for this, such as G/A per 90, etc.)
    - Which goalkeeper was most responsible for their team's success? (we can calculate some statistics for this, such as saves per 90, etc.)
- As the rounds go on, do teams tend to play more conservatively, or do they play more aggressively? (this can be measured by shots attempted, possession, etc.)
- Having possession is an emphasis in modern football, but does having more possession actually lead to more success?

1.1 Importing packages

```
[ ]: %matplotlib inline
     import warnings
     warnings.simplefilter(action='ignore')

     import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     import plotly.express as px
     from plotnine import *
```

1.2 Loading, cleaning, and exploring datasets

We have multiple main datasets which we will be working with in our project; our main dataset is the `matches` dataset which contains information about the matches played in the 2022 Qatar World Cup. The second dataset, `countries`, relates to the individual country statistics, which we will be aggregating from our information about each match in order to get a better understanding of the teams and their performance. Lastly, we are going to take a look at a `players` dataset, which contains much information about the actual participants in the tournament, and their performance in the matches.

1.2.1 Matches Dataset - loading and cleaning

```
[ ]: matches = pd.read_csv('cup.csv')
     matches[['team1', 'team2']].head()
```

```
[ ]:           team1         team2
     0         QATAR       ECUADOR
     1       ENGLAND          IRAN
     2       SENEGAL   NETHERLANDS
     3  UNITED STATES        WALES
     4     ARGENTINA  SAUDI ARABIA
```

```
[ ]: matches = matches[[
         'team1',
         'team2',
         'possession team1',
         'possession team2',
         'possession in contest',
         'number of goals team1',
         'number of goals team2',
         'category',
         'total attempts team1',
         'total attempts team2',
         'conceded team1',
         'conceded team2',
         'goal inside the penalty area team1',
         'goal inside the penalty area team2',
         'goal outside the penalty area team1',
         'goal outside the penalty area team2',
         'assists team1',
         'assists team2',
         'yellow cards team1',
         'yellow cards team2',
         'red cards team1',
         'red cards team2',
         'fouls against team1',
         'fouls against team2',
         'passes team1',
         'passes team2',
         'passes completed team1',
         'passes completed team2',
         'penalties scored team1',
         'penalties scored team2',
         'goal preventions team1',
         'goal preventions team2',
         'forced turnovers team1',
         'forced turnovers team2'
     ]]

     matches['team1'] = matches['team1'].str.title()
     matches['team2'] = matches['team2'].str.title()

     matches['GroupID'] = matches['category'].apply(lambda x: x[-1] if 'Group' in x␣
      ↪else np.nan)
     matches['category'] = matches['category'].apply(lambda x: x[:-2] if 'Group' in␣
      ↪x else x)

     matches['possession team1'] = matches['possession team1'].str[:-1].astype(int)
     matches['possession team2'] = matches['possession team2'].str[:-1].astype(int)
     matches['possession in contest'] = matches['possession in contest'].str[:-1].
      ↪astype(int)
```

```
[ ]: matches['category'].unique()
```

```
[ ]: array(['Group', 'Round of 16', 'Quarter-final', 'Semi-final',
            'Play-off for third place', 'Final'], dtype=object)
```

```
[ ]: # dividing the data based on what stage of the tournament the match was in␣
      ↪(just in case we want to look at the data in this way later)
     groupMatches = matches.loc[matches['category'] == 'Group']
     ro16 = matches.loc[matches['category'] == 'Round of 16']
```

```
quarterfinals = matches.loc[matches['category'] == 'Quarter-final']
semifinals = matches.loc[matches['category'] == 'Semi-final']
thirdPlaceMatch = matches.loc[matches['category'] == 'Play-off for third place']
final = matches.loc[matches['category'] == 'Final']
```

1.2.2 Countries Dataset - loading and cleaning

[ ]:

```
countries = pd.read_csv('precup_rank.csv', index_col='Nation')
countries.head()
```

[ ]:

```
           Rank   Points  World Cup Wins
Nation
Brazil        1  1841.30               5
Belgium       2  1816.71               0
Argentina     3  1773.88               3
France        4  1759.78               2
England       5  1728.47               1
```

[ ]:

```
# we are creating a lot of columns, and it doesn't look pretty, but this is␣
 ↪what we want to do; is there a better way to do this?
# also, a problem that might arise is that it will become hard to distinguish␣
 ↪actual zeroes from missing values, which we might need to research a better␣
 ↪way for that as well
countries['Possession in Group'] = 0
countries['Possession in Round of 16'] = 0
countries['Possession in Quarter-final'] = 0
countries['Possession in Semi-final'] = 0
countries['Possession in Play-off for third place'] = 0
countries['Possession in Final'] = 0

countries['Total Passes in Group'] = 0
countries['Total Passes in Round of 16'] = 0
countries['Total Passes in Quarter-final'] = 0
countries['Total Passes in Semi-final'] = 0
countries['Total Passes in Play-off for third place'] = 0
countries['Total Passes in Final'] = 0

countries['Completed Passes in Group'] = 0
countries['Completed Passes in Round of 16'] = 0
countries['Completed Passes in Quarter-final'] = 0
countries['Completed Passes in Semi-final'] = 0
countries['Completed Passes in Play-off for third place'] = 0
countries['Completed Passes in Final'] = 0

countries['Shot Attempts in Group'] = 0
countries['Shot Attempts in Round of 16'] = 0
```

```
countries['Shot Attempts in Quarter-final'] = 0
countries['Shot Attempts in Semi-final'] = 0
countries['Shot Attempts in Play-off for third place'] = 0
countries['Shot Attempts in Final'] = 0

countries['Total Goals in Group'] = 0
countries['Total Goals in Round of 16'] = 0
countries['Total Goals in Quarter-final'] = 0
countries['Total Goals in Semi-final'] = 0
countries['Total Goals in Play-off for third place'] = 0
countries['Total Goals in Final'] = 0

countries['Close Range Goals in Group'] = 0
countries['Close Range Goals in Round of 16'] = 0
countries['Close Range Goals in Quarter-final'] = 0
countries['Close Range Goals in Semi-final'] = 0
countries['Close Range Goals in Play-off for third place'] = 0
countries['Close Range Goals in Final'] = 0

countries['Long Range Goals in Group'] = 0
countries['Long Range Goals in Round of 16'] = 0
countries['Long Range Goals in Quarter-final'] = 0
countries['Long Range Goals in Semi-final'] = 0
countries['Long Range Goals in Play-off for third place'] = 0
countries['Long Range Goals in Final'] = 0

countries['Conceded in Group'] = 0
countries['Conceded in Round of 16'] = 0
countries['Conceded in Quarter-final'] = 0
countries['Conceded in Semi-final'] = 0
countries['Conceded in Play-off for third place'] = 0
countries['Conceded in Final'] = 0

countries['Assists in Group'] = 0
countries['Assists in Round of 16'] = 0
countries['Assists in Quarter-final'] = 0
countries['Assists in Semi-final'] = 0
countries['Assists in Play-off for third place'] = 0
countries['Assists in Final'] = 0

countries['Fouls in Group'] = 0
countries['Fouls in Round of 16'] = 0
countries['Fouls in Quarter-final'] = 0
countries['Fouls in Semi-final'] = 0
countries['Fouls in Play-off for third place'] = 0
countries['Fouls in Final'] = 0
```

```
countries['Yellow Cards in Group'] = 0
countries['Yellow Cards in Round of 16'] = 0
countries['Yellow Cards in Quarter-final'] = 0
countries['Yellow Cards in Semi-final'] = 0
countries['Yellow Cards in Play-off for third place'] = 0
countries['Yellow Cards in Final'] = 0

countries['Red Cards in Group'] = 0
countries['Red Cards in Round of 16'] = 0
countries['Red Cards in Quarter-final'] = 0
countries['Red Cards in Semi-final'] = 0
countries['Red Cards in Play-off for third place'] = 0
countries['Red Cards in Final'] = 0

countries['Saves in Group'] = 0
countries['Saves in Round of 16'] = 0
countries['Saves in Quarter-final'] = 0
countries['Saves in Semi-final'] = 0
countries['Saves in Play-off for third place'] = 0
countries['Saves in Final'] = 0

countries['Penalties in Group'] = 0
countries['Penalties in Round of 16'] = 0
countries['Penalties in Quarter-final'] = 0
countries['Penalties in Semi-final'] = 0
countries['Penalties in Play-off for third place'] = 0
countries['Penalties in Final'] = 0

countries['Own Goals in Group'] = 0
countries['Own Goals in Round of 16'] = 0
countries['Own Goals in Quarter-final'] = 0
countries['Own Goals in Semi-final'] = 0
countries['Own Goals in Play-off for third place'] = 0
countries['Own Goals in Final'] = 0

countries['Forced Turnovers in Group'] = 0
countries['Forced Turnovers in Round of 16'] = 0
countries['Forced Turnovers in Quarter-final'] = 0
countries['Forced Turnovers in Semi-final'] = 0
countries['Forced Turnovers in Play-off for third place'] = 0
countries['Forced Turnovers in Final'] = 0

countries['GroupID'] = ''
```

Now that we have our `countries` dataset prepared for data to enter it, we need to start to modify our `matches` dataset, so that any redundant information is discarded, and all the information we need is correctly represented.

We are now going to write some code which will allow us to clean up the way some of this data looks. Ideally, we want to observe these statistics based on country, while we have it here as `team1` or `team2`, which isn't really helpful if we want to get a context of a particular country. So, using the `countries` dataset that we have introduced earlier that just contains their FIFA rank at the time of the World Cup, we will be adding each countries individual statistics to the dataset.

[ ]:

```
def parseRound(round):
    teamID = ''
    for team in countries.index:
        roundMatches = round.loc[((round['team1'] == team) | (round['team2'] ==␣
 ↪team))]
        for _, match in roundMatches.iterrows():

            if match['team1'] == team:
                teamID = '1'
            else:
                teamID = '2'

            if match['category'] == 'Group':
                countries.loc[team, f'GroupID'] = match[f'GroupID']

            #print(team, match[f'number of goals team{teamID}'])
            countries.loc[team, f'Possession in {match["category"]}'] +=␣
 ↪match[f'possession team{teamID}']
            countries.loc[team, f'Total Passes in {match["category"]}'] +=␣
 ↪match[f'passes team{teamID}']
            countries.loc[team, f'Completed Passes in {match["category"]}'] +=␣
 ↪match[f'passes completed team{teamID}']
            countries.loc[team, f'Total Goals in {match["category"]}'] +=␣
 ↪match[f'number of goals team{teamID}']
            countries.loc[team, f'Conceded in {match["category"]}'] +=␣
 ↪match[f'conceded team{teamID}']
            countries.loc[team, f'Assists in {match["category"]}'] +=␣
 ↪match[f'assists team{teamID}']
            countries.loc[team, f'Forced Turnovers in {match["category"]}'] +=␣
 ↪match[f'forced turnovers team{teamID}']
            countries.loc[team, f'Saves in {match["category"]}'] +=␣
 ↪match[f'goal preventions team{teamID}']
            countries.loc[team, f'Penalties in {match["category"]}'] +=␣
 ↪match[f'penalties scored team{teamID}']
            countries.loc[team, f'Fouls in {match["category"]}'] +=␣
 ↪match[f'fouls against team{teamID}']
            countries.loc[team, f'Yellow Cards in {match["category"]}'] +=␣
 ↪match[f'yellow cards team{teamID}']
            countries.loc[team, f'Red Cards in {match["category"]}'] +=␣
 ↪match[f'red cards team{teamID}']
```

```
            countries.loc[team, f'Shot Attempts in {match["category"]}'] +=␣
 ↪match[f'total attempts team{teamID}']


parseRound(groupMatches)
parseRound(ro16)
parseRound(quarterfinals)
parseRound(semifinals)
parseRound(thirdPlaceMatch)
parseRound(final)
```

```
[ ]: countries['Average Possession in Group'] = round(countries['Possession in␣
      ↪Group'] / 3, 2)
     countries['Goals Per Game in Group'] = round(countries['Total Goals in Group'] /
      ↪ 3, 2)

     countries['Total Goals'] = countries['Total Goals in Group'] + countries['Total␣
      ↪Goals in Round of 16'] + \
         countries['Total Goals in Quarter-final'] + countries['Total Goals in␣
      ↪Semi-final'] + countries['Total Goals in Play-off for third place'] \
             + countries['Total Goals in Final']
```

```
[ ]: groupA = countries.loc[countries['GroupID'] == 'A']
     groupB = countries.loc[countries['GroupID'] == 'B']
     groupC = countries.loc[countries['GroupID'] == 'C']
     groupD = countries.loc[countries['GroupID'] == 'D']
     groupE = countries.loc[countries['GroupID'] == 'E']
     groupF = countries.loc[countries['GroupID'] == 'F']
     groupG = countries.loc[countries['GroupID'] == 'G']
     groupH = countries.loc[countries['GroupID'] == 'H']

     print(f'Group A average team ranking: {groupA["Rank"].mean()}')
     print(f'Group B average team ranking: {groupB["Rank"].mean()}')
     print(f'Group C average team ranking: {groupC["Rank"].mean()}')
     print(f'Group D average team ranking: {groupD["Rank"].mean()}')
     print(f'Group E average team ranking: {groupE["Rank"].mean()}')
     print(f'Group F average team ranking: {groupF["Rank"].mean()}')
     print(f'Group G average team ranking: {groupG["Rank"].mean()}')
     print(f'Group H average team ranking: {groupH["Rank"].mean()}')
```

```
Group A average team ranking: 30.0
Group B average team ranking: 15.0
Group C average team ranking: 23.25
Group D average team ranking: 20.5
Group E average team ranking: 18.25
Group F average team ranking: 19.25
Group G average team ranking: 20.0
Group H average team ranking: 28.0
```

It seems that based on the mean ranks of our groups, Group B has the highest average rank, which coins this group 'The Group of Death', a term designated to the toughest group in the tournament. On the other hand, Group A seems to have the lowest average rank. Let's take a look at the two groups, and see what that's about.

```
[ ]: groupB[['Rank']]
```

```
[ ]:                Rank
     Nation
     England           5
     United States    16
     Wales            19
     Iran             20
```

Looking at the group of death, we can see that all 4 teams are ranked in the top 20, with 3 of the teams being very close to each other in rank.

```
[ ]: groupA[['Rank']]
```

```
[ ]:              Rank
     Nation
     Netherlands     8
     Senegal        18
     Ecuador        44
     Qatar          50
```

This group is much different than Group B; while we can see that we have 2 top 20 nations here, but two of the lower ranked countries in the tournaments as well, in Ecuador and Qatar (the host country)

1.2.3 Players Dataset - loading and cleaning

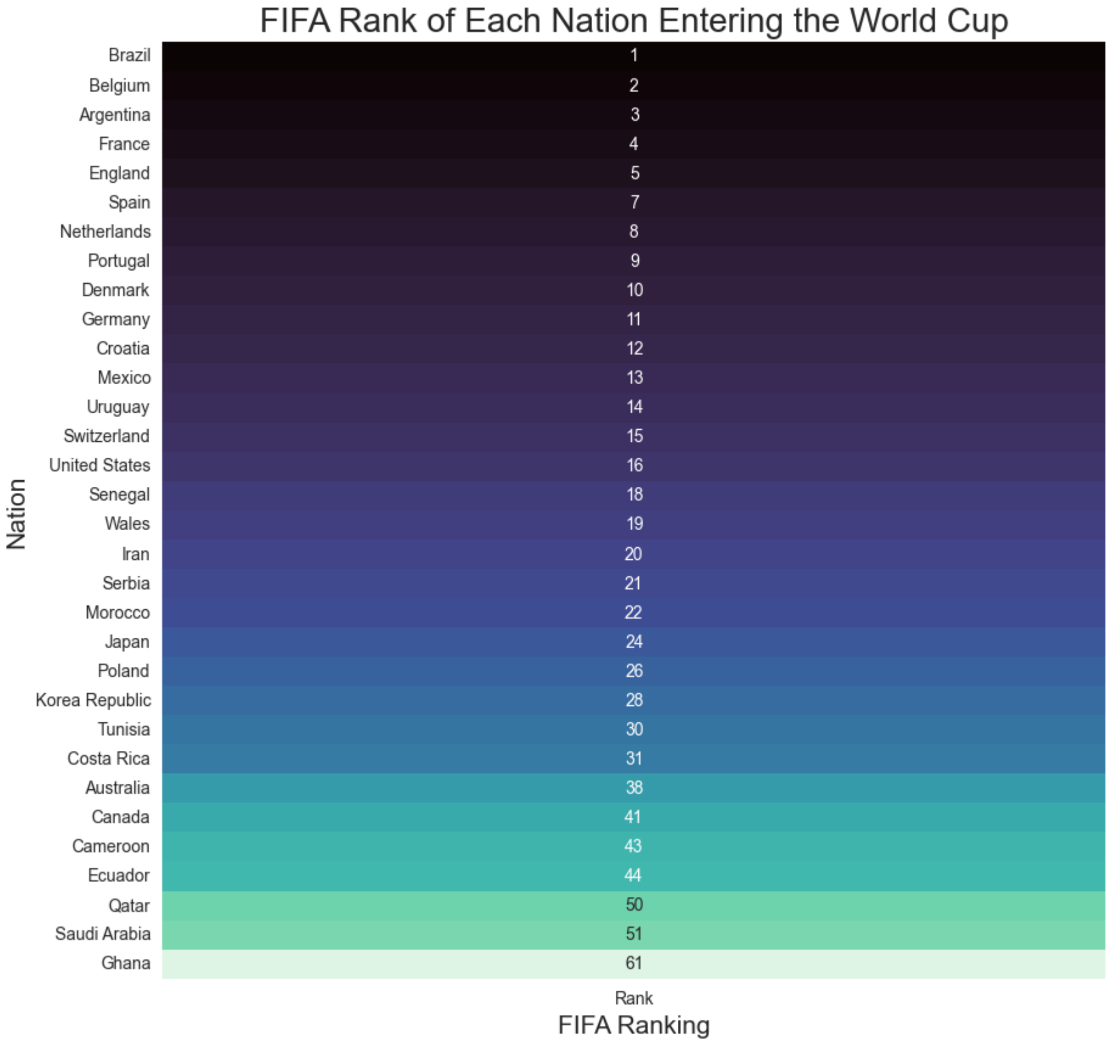
```
[ ]: players = pd.read_csv('player_stats.csv')

     duplicate_columns = ['player', 'club', 'position', 'age', 'team', 'birth_year',␣
       ↪'minutes_90s']
     players = players[duplicate_columns]

     keeper_columns =␣
       ↪['position','team','age','club','birth_year','games','games_starts','minutes','minutes_90s'

     player_defense = pd.read_csv('player_defense.csv').drop(duplicate_columns[2:],␣
       ↪axis=1)
     player_shooting = pd.read_csv('player_shooting.csv').drop(duplicate_columns[2:
       ↪], axis=1)
```

```
player_stats = pd.read_csv('player_stats.
 ↪csv')[['player','xg_per90','xg_assist_per90','xg_xg_assist_per90','npxg_per90','npxg_xg_ass
player_possession = pd.read_csv('player_possession.csv').
 ↪drop(duplicate_columns[2:], axis=1)
player_keepers = pd.read_csv('player_keepers.csv').drop(keeper_columns, axis=1)
player_passing = pd.read_csv('player_passing.csv').drop(duplicate_columns[2:],␣
 ↪axis=1)
player_playingtime = pd.read_csv('player_playingtime.csv')[['player',␣
 ↪'minutes','minutes_per_game']]
player_gca = pd.read_csv('player_gca.csv')[['player',␣
 ↪'sca','sca_per90','gca','gca_per90']]

player_keepers = pd.merge(players.loc[players['position'] == 'GK'],␣
 ↪player_keepers, on='player', how='left')


# merging all of the above dataframes into one
players = players.merge(player_defense, on='player')
players = players.merge(player_shooting, on='player')
players = players.merge(player_possession, on='player')
players = players.merge(player_passing, on='player')
players = players.merge(player_stats, on='player')
players = players.merge(player_playingtime, on='player')
players = players.merge(player_gca, on='player')

players['age'] = players['age'].astype(str).str[:2].astype(int)
players['G/A'] = players['goals'] + players['assists']

players[['player', 'team']].head()
```

```
[ ]:                 player       team
     0            Aaron Mooy  Australia
     1          Aaron Ramsey      Wales
     2     Abdelhamid Sabiri    Morocco
     3     Abdelkarim Hassan      Qatar
     4  Abderrazak Hamdallah    Morocco
```

1.3 Background analysis: Visualizing the Countries

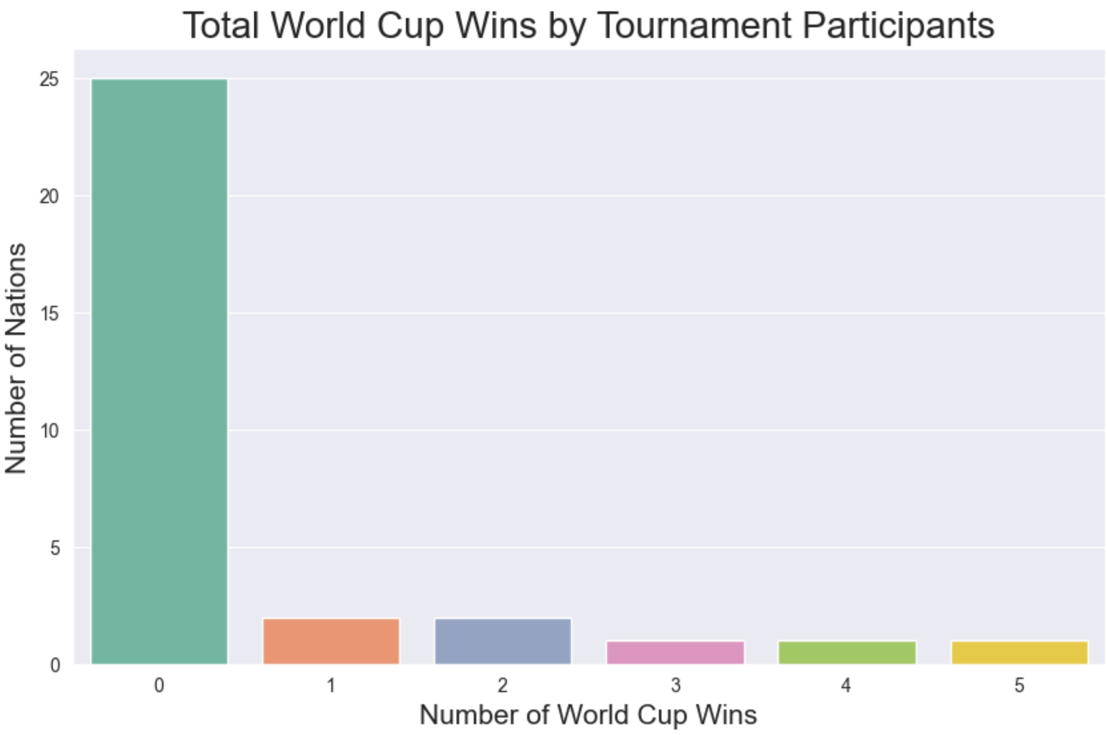
```
[ ]: # maybe change this visual?
     plt.figure(figsize=(10, 10))
     sns.set_style('darkgrid')

     sns.heatmap(countries['Rank'].to_frame(), annot=True, fmt='g', cmap='mako',␣
      ↪cbar=False)
     plt.title('FIFA Rank of Each Nation Entering the World Cup', fontsize=20)
     plt.xlabel('FIFA Ranking', fontsize=15)
     plt.ylabel('Nation', fontsize=15)
     plt.show()
```



This is interesting to visualize because one might assume that a tournament consisting of the best 32 national teams in the world, would have the top 32 ranked countries participating in the tournament. However, this is not the case, as qualification to the World Cup is a long and difficult process, and many of the top ranked countries in the world are not able to qualify for the tournament. This is why we have countries like Saudi Arabia and Ghana, who are ranked 51st and 61st in the world respectively, participating in the tournament.

[ ]:
```
plt.figure(figsize=(10, 6))
sns.set_style('darkgrid')

sns.countplot(x='World Cup Wins', data=countries, palette='Set2')
```

```
plt.title('Total World Cup Wins by Tournament Participants', fontsize=20)
plt.xlabel('Number of World Cup Wins', fontsize=15)
plt.ylabel('Number of Nations', fontsize=15)
plt.show()
```



It is easy here to see that most of the countries have never won a World Cup, and a few countries have won one, or two. There are three countries however that have won 3 or more, so let's see who they are.

```
[ ]: countries[countries['World Cup Wins'] >= 3]['World Cup Wins'].to_frame().
       ↪sort_values('World Cup Wins', ascending=False)
```

```
[ ]:             World Cup Wins
     Nation
     Brazil                   5
     Germany                  4
     Argentina                3
```

```
[ ]: TOTAL_WORLD_CUPS = 22
     TOTAL_WORLD_CUPS - countries['World Cup Wins'].sum()
```

```
[ ]: 4
```

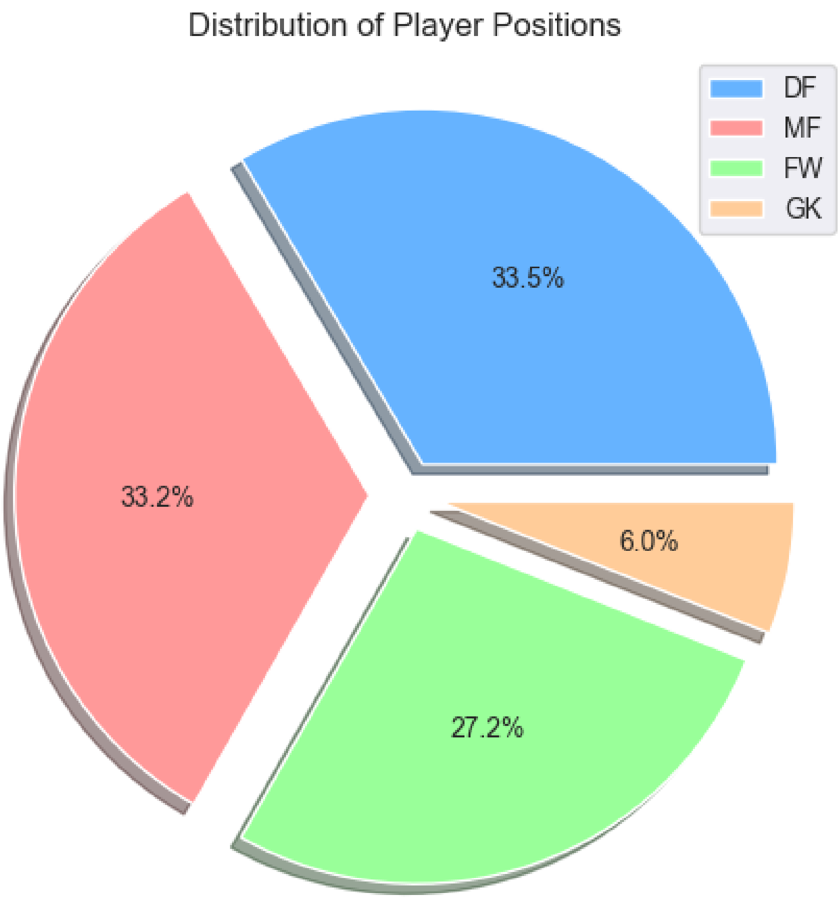
This is notable because the total number of World Cups played is 22 (counting this one), and we have 18 of the winners represented here. That would mean that 4 World Cup winners did not qualify for this World Cup, which is a very interesting statistic. The other 4 missing World Cups were all won by Italy, who failed to qualify to this World Cup due to a surprise 1-0 upset loss in a World Cup qualifying match to North Macedonia.

1.4 Visualizing the players, and their performances

Background Analysis Some things we want to see with our players:

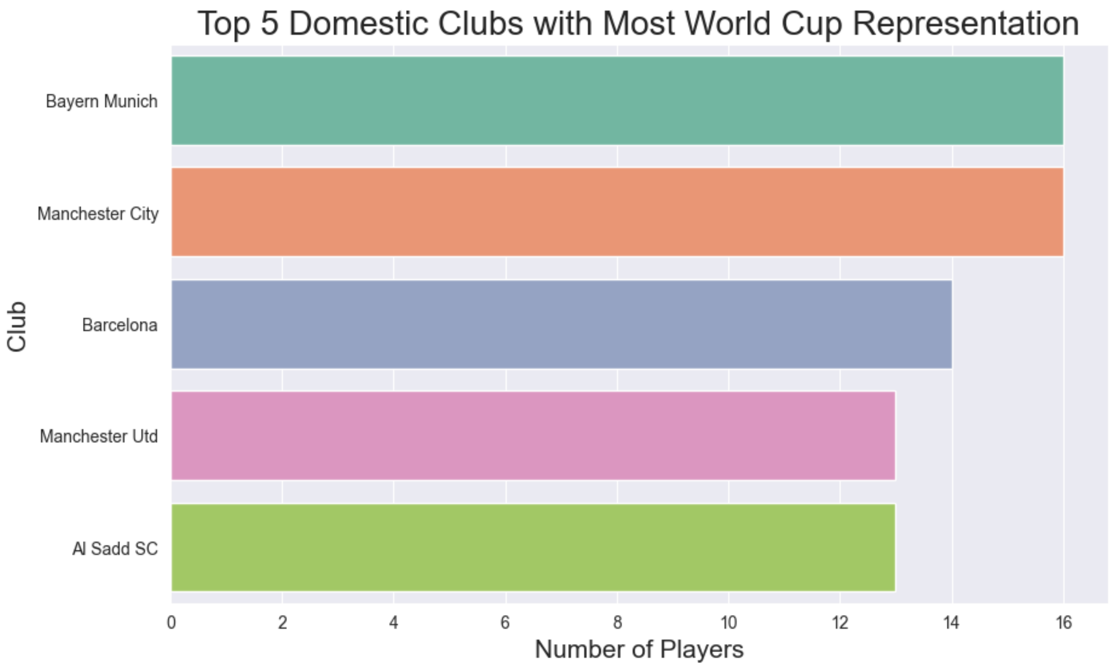
- Most represented clubs
- Distribution of positions
- Average age of players
- Top goal scorers
- Top assisters
- Top players under 21 goals + assists (g/a)

[ ]:
```
colors = ['#66b3ff','#ff9999','#99ff99','#ffcc99']

ax = plt.figure(figsize=(10, 6))

ax = players['position'].value_counts().plot(kind='pie',
                                          autopct='%1.1f%%', shadow=True,␣
  ↪explode=[0.1, 0.1, 0.1, 0.1], colors=colors,
                                          legend=True, title='Distribution of␣
  ↪Player Positions', ylabel='', labeldistance=None)
```



Here we can see a very similar distribution of defenders and midfielders, which makes sense because there are multiple defenders and midfielders in team compositions, and only one goalkeeper and usually one or two forwards.

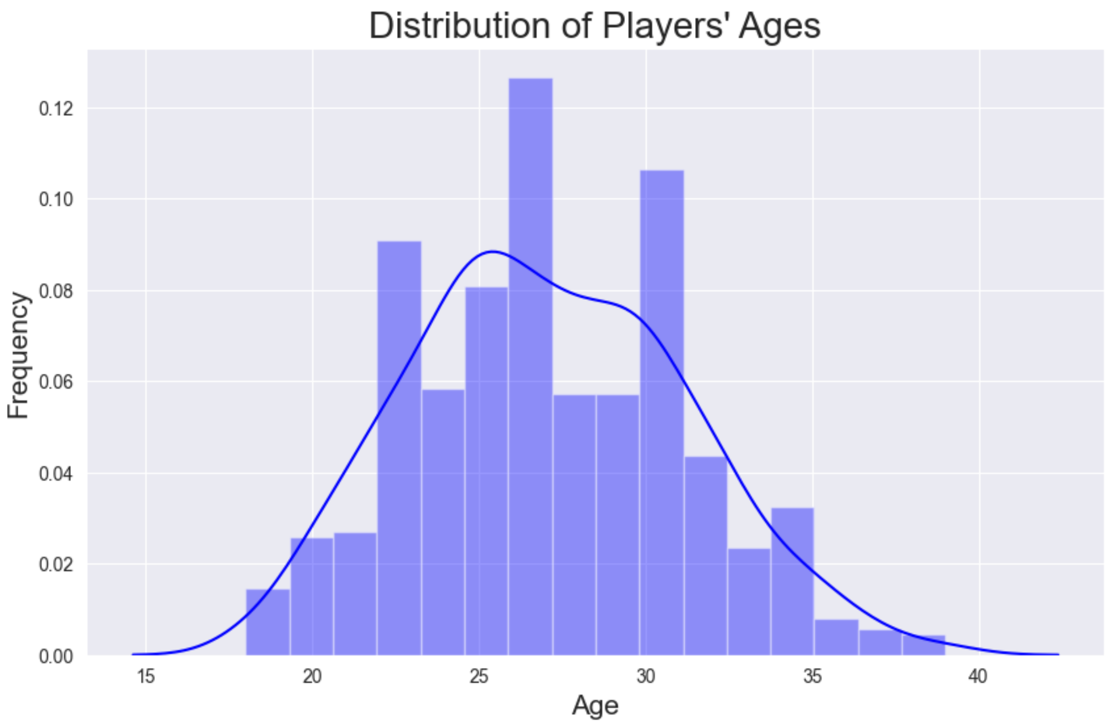
```
[ ]: plt.figure(figsize=(10, 6))
     sns.set_style('darkgrid')
     sns.countplot(y='club', data=players, order=players['club'].value_counts().
       ↪iloc[:5].index, palette='Set2')
     plt.title('Top 5 Domestic Clubs with Most World Cup Representation',␣
       ↪fontsize=20)
     plt.xlabel('Number of Players', fontsize=15)
     plt.ylabel('Club', fontsize=15)
     plt.show()
```



Al Sadd SC is an interesting outlier because all of these other clubs are very well known and highly regarded around the world, but Al Sadd SC does not fit the mold of the rest of these historic clubs. The reason behind the strong showing from Al Saad SC is because it is a club in Qatar (who is the host nation of this World Cup), and the majority of the players from the Qatar national team actually play for that club.

[ ]:

```
plt.figure(figsize=(10, 6))
sns.set_style('darkgrid')
sns.distplot(players['age'], color='blue')
plt.title('Distribution of Players\' Ages', fontsize=20)
plt.xlabel('Age', fontsize=15)
plt.ylabel('Frequency', fontsize=15)
plt.show()
```



```
[ ]: players['age'].mean()
```

```
[ ]: 27.054411764705883
```

Based on the visual, and with our .mean() as further proof, we can see that there are a lot of players in the 27 yera old range, with other spikes at around the 30-32 range, and at the 21-23 range as well. This makes us interested, though, in who the oldest and youngest players were in the tournament.

```
[ ]: players.sort_values('age', ascending=False).head()[['player', 'age', 'team',␣
      ↪'position', 'goals', 'assists']].sort_values('goals', ascending=False)
```

```
[ ]:                  player  age        team position  goals  assists
     519                Pepe   39    Portugal       DF      1        0
     142          Dani Alves   39      Brazil       DF      0        0
     75    Atiba Hutchinson   39      Canada       MF      0        0
     616        Thiago Silva   38      Brazil       DF      0        1
     108          Bryan Ruiz   37  Costa Rica       MF      0        0
```

```
[ ]: players.sort_values('age').head()[['player', 'age', 'team', 'position',␣
      ↪'goals', 'assists']].sort_values('goals', ascending=False)
```

```
[ ]:                        player  age       team position  goals  assists
     217                      Gavi   18      Spain       MF      1        0
```

```
215           Garang Kuol   18  Australia       FW      0         0
666       Youssoufa Moukoko   18    Germany       FW      0         0
94         Bilal El Khannous   18    Morocco       MF      0         0
7     Abdul Fatawu Issahaku   18      Ghana       FW      0         0
```

It was an interesting observation that there are no players in the tournament actually over the age of 40, and that there were no players under the age of 18.

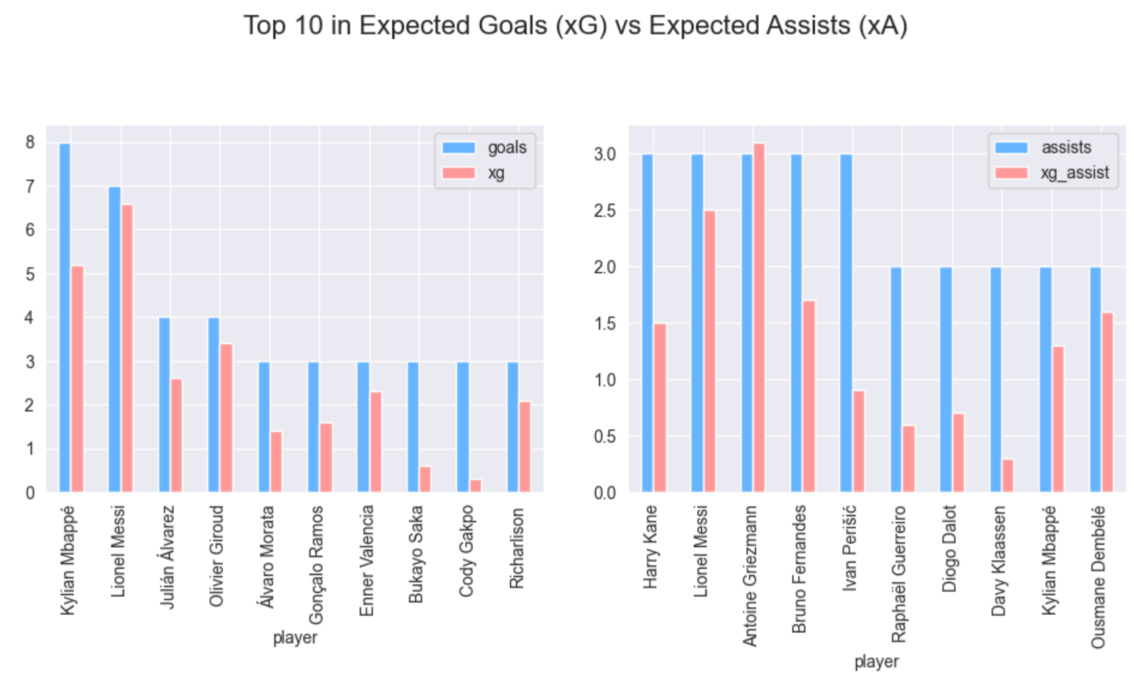
1.4.1 Advanced Statistics

In doing research for this project, it has come to our attention that there are a certain selection of advanced statistics that are considered to be relevant in regard to player performance. The dataset which we are working with *does* contain these statistics, so we are going to take a look at them, and see if we can find any interesting trends or insights.

Observing xG and xA xG, or Expected Goals, is a metric that is used to measure the quality of a shot taken by a player. The higher the xG, the more likely the shot is to go in. xA, or Expected Assists, is a metric that is used to measure the quality of a pass that leads to a shot. The higher the xA, the more likely the pass is to lead to a goal. These are key metrics in modern football, and are frequently used to measure the quality of a player's performance. We will be exploring these metrics to determine which players are deemed most efficient in the tournament based on their criteria.

[ ]:

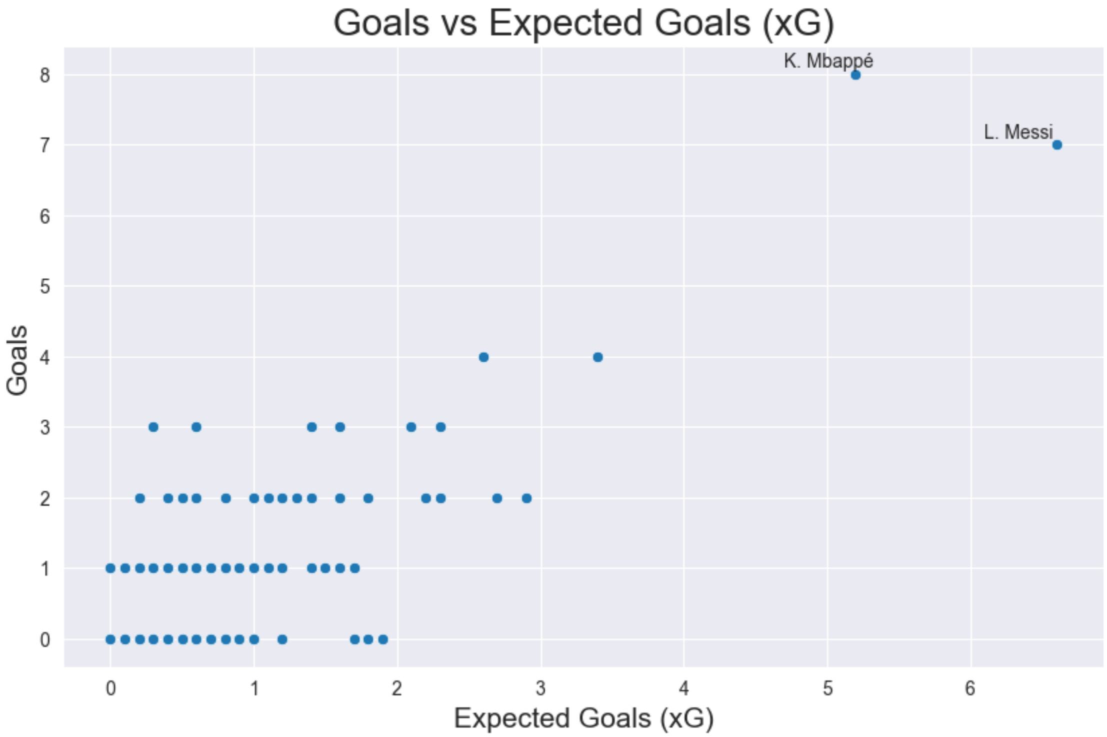
```python
# fix this; kind of ugly
_top_xg = players.sort_values('goals', ascending=False).head(10)
_top_xa = players.sort_values('assists', ascending=False).head(10)
top_xg = _top_xg[['player', 'goals', 'xg']]
top_xassist = _top_xa[['player', 'assists', 'xg_assist']]
top_xg = top_xg.set_index('player')
top_xassist = top_xassist.set_index('player')



fig, axes = plt.subplots(1, 2, figsize=(15, 5))
sns.set_style('darkgrid')

top_xg.plot.bar(figsize=(10, 6), color=['#66b3ff','#ff9999'], ax=axes[0])
top_xassist.plot.bar(figsize=(10, 6), color=['#66b3ff','#ff9999'], ax=axes[1])

fig.suptitle('Top 10 in Expected Goals (xG) vs Expected Assists (xA)',␣
 ↪fontsize=16)
fig.tight_layout(pad=3.0)
```


Top 10 in Expected Goals (xG) vs Expected Assists (xA)

By observing xG, we can observe that someone like Lionel Messi, scored around the same amount of goals as his xG, meaning that he was statistically scoring all the goals that he was expected to. Players like Bukayo Saka and Cody Gakpo both had relatively low xG (under 1 xG), but finished with 3 goals each, so they massively outperformed their xG.

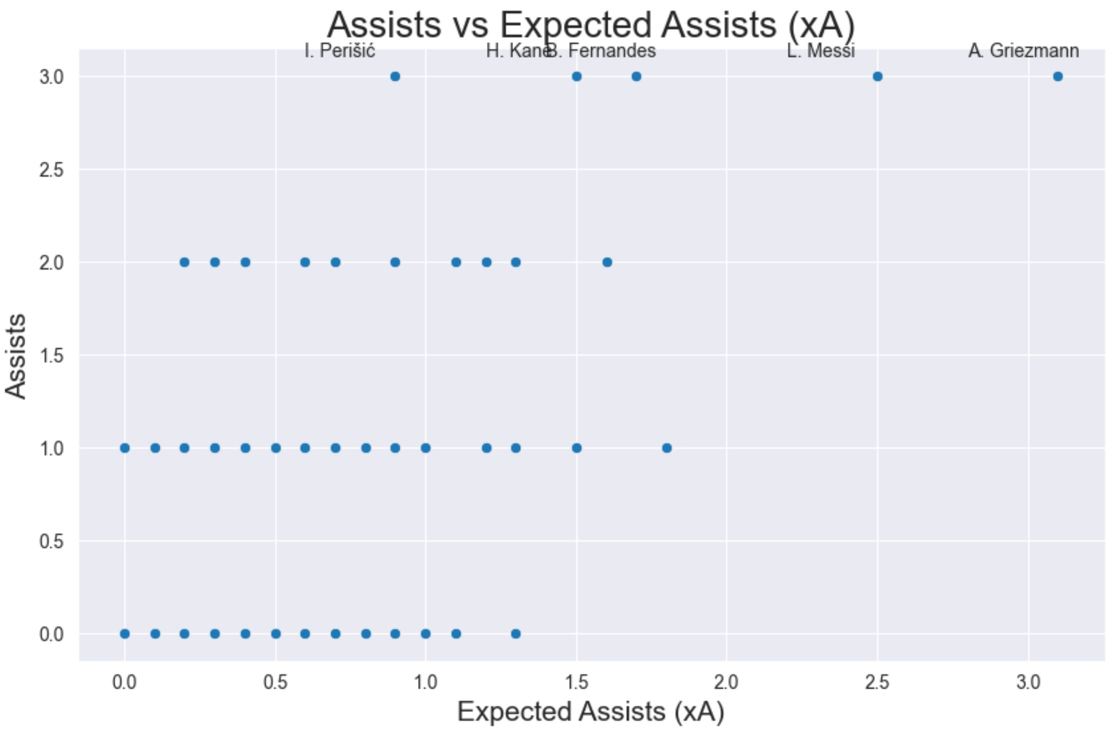
On the other hand, with xA, we can see that a player like Ivan Perisic was able to get 3 assists, but with an xA of a little under 1, meaning that he was able to outperform his xA by a lot. Inversely, Antoine Griezmann had an xA that was actually *higher* than his output of assists, meaning that his quality of passing was better than the players who were finishing his passes.

[ ]:
```
plt.figure(figsize=(10, 6))
sns.set_style('darkgrid')

sns.scatterplot(x='xg', y='goals', data=players, palette='Set3')
plt.title('Goals vs Expected Goals (xG)', fontsize=20)
plt.xlabel('Expected Goals (xG)', fontsize=15)
plt.ylabel('Goals', fontsize=15)

# finding the outliers
outliers = players[players['goals'] > players['goals'].quantile(.97) + 1.5 *␣
  ↪(players['goals'].quantile(.97) - players['goals'].quantile(.03))]


# adding labels to the outliers
for i, row in outliers.iterrows():
    playerName = row['player']
    try:
        formattedPlayerName = f'{playerName[0]}. {playerName.split(" ")[1]}'
    except:
        # they only go by one name: ex. Richarlison or Gavi
        formattedPlayerName = playerName
    plt.annotate(formattedPlayerName, xy=(row['xg'], row['goals']),␣
 ↪xytext=(row['xg']-.5, row['goals']+.1))
```



[ ]:

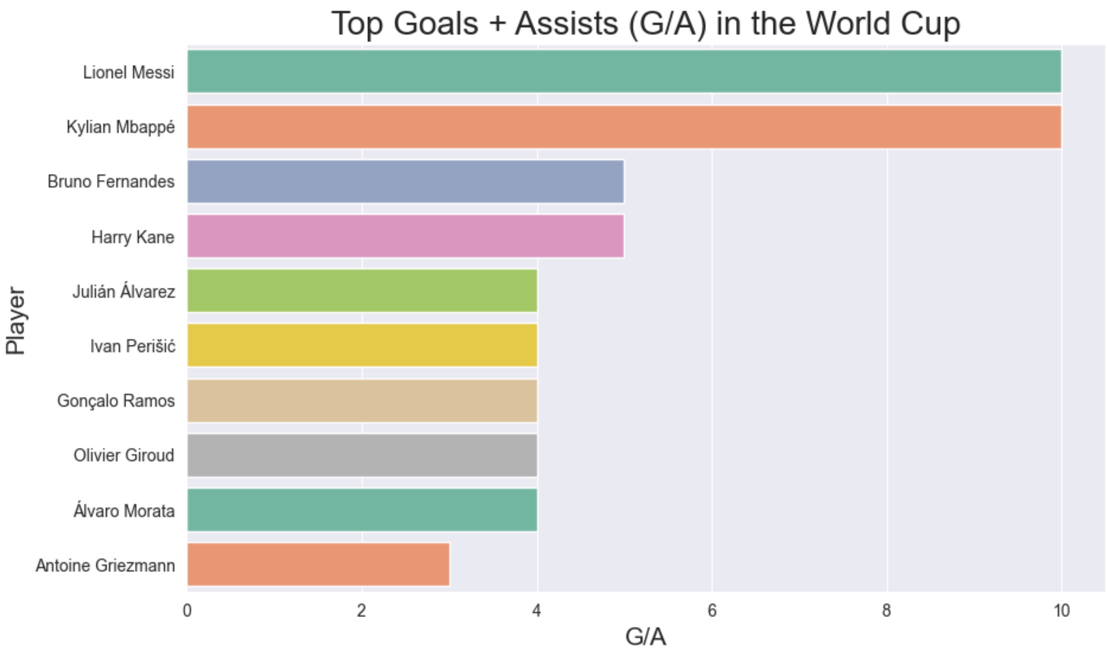
```
plt.figure(figsize=(10, 6))
sns.set_style('darkgrid')

sns.scatterplot(x='xg_assist', y='assists', data=players, palette='Set3')
plt.title('Assists vs Expected Assists (xA)', fontsize=20)
plt.xlabel('Expected Assists (xA)', fontsize=15)
plt.ylabel('Assists', fontsize=15)

# Find the outliers
outliers = players[players['assists'] > players['assists'].quantile(.96) + 1.5␣
 ↪* (players['assists'].quantile(.96) - players['assists'].quantile(0.04))]


# Add labels to the outliers
for i, row in outliers.iterrows():
    playerName = row['player']
    try:
        formattedPlayerName = f'{playerName[0]}. {playerName.split(" ")[1]}'
    except:
        # they only go by one name: ex. Richarlison or Gavi
        formattedPlayerName = playerName
    plt.annotate(formattedPlayerName, xy=(row['xg_assist'], row['assists']),␣
  ↪xytext=(row['xg_assist']-0.3, row['assists']+0.1))
```



Observing G/A and G/A per 90 G/A is a metric that is used to measure the total amount of goals and assists that a player has, by simply adding together the two. G/A per 90 is a metric that is used to measure the total amount of goals and assists that a player has per 90 minutes played. We will be exploring these metrics to determine which players are deemed most efficient in the tournament based on these criteria.

```
[ ]: plt.figure(figsize=(10, 6))
     sns.set_style('darkgrid')

     sns.barplot(x='G/A', y='player', data=players.sort_values('G/A',␣
       ↪ascending=False).head(10), palette='Set2')
     plt.title('Top Goals + Assists (G/A) in the World Cup', fontsize=20)
     plt.xlabel('G/A', fontsize=15)
     plt.ylabel('Player', fontsize=15)
```

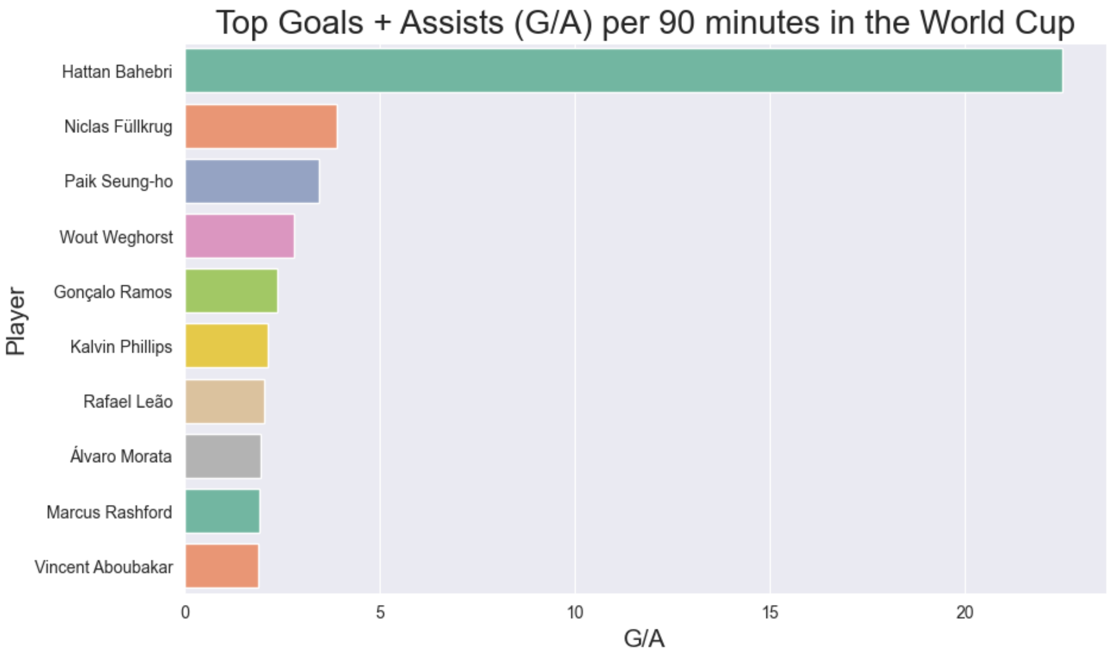
```
[ ]: Text(0, 0.5, 'Player')
```



Ultimately, an easy indicator of a player's performance is their goals + assists (G/A), and we can see that tied at the top, is none other than Lionel Messi and Kylian Mbappe, who both had a total of 10 G/A throughout the tournament. Again, it's important to note that both Messi and Mbappe played in every possible match in the tournament, which can serve as a reason why their G/A output is so high compared to others. This is why `G/A per 90` is a statistic that is used to measure the efficiency of a player, because it takes into account the amount of time that a player is on the pitch, and how much they are able to produce in that time.

```
[ ]: plt.figure(figsize=(10, 6))
     sns.set_style('darkgrid')

     sns.barplot(x='goals_assists_per90', y='player', data=players.
       ↪sort_values('goals_assists_per90', ascending=False).head(10), palette='Set2')
     plt.title('Top Goals + Assists (G/A) per 90 minutes in the World Cup',␣
       ↪fontsize=20)
     plt.xlabel('G/A', fontsize=15)
     plt.ylabel('Player', fontsize=15)
```

```
[ ]: Text(0, 0.5, 'Player')
```



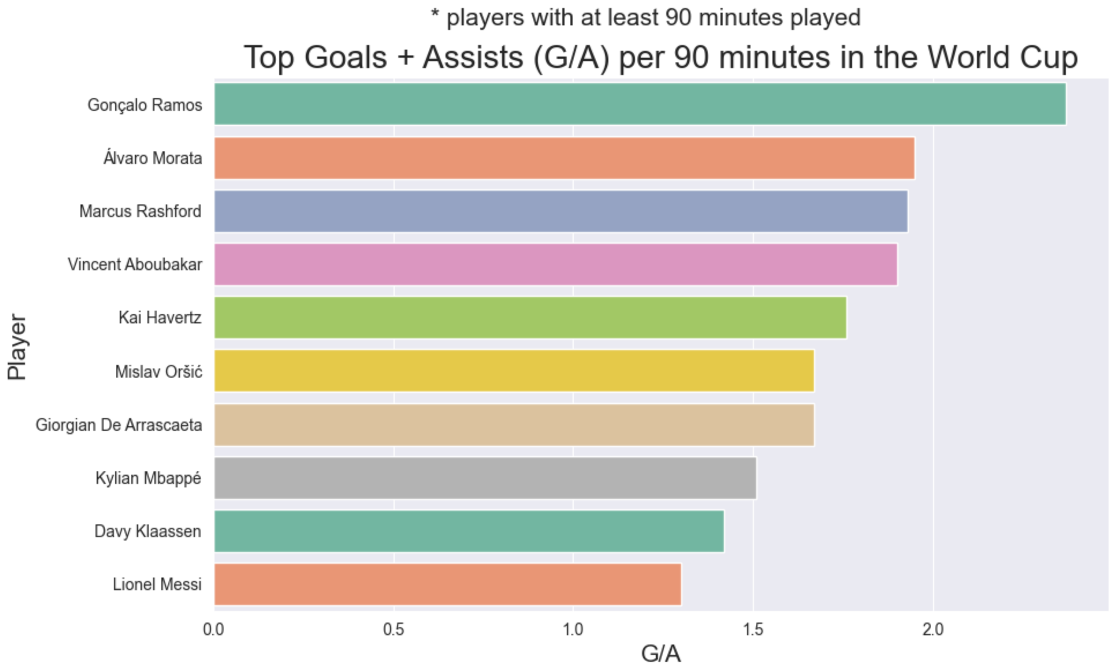
```
[ ]: players[players['player'] == 'Hattan Bahebri'][['goals', 'assists', 'minutes',␣
       ↪'goals_assists_per90']]
```

```
[ ]:      goals  assists  minutes  goals_assists_per90
     246      0        1      4.0                 22.5
```

We have a very humorous outlier here, when we are looking per 90; Hattan Behebri, who had played 4 minutes all tournament, actually ended up with an assist in those 4 minutes, thus giving him a very high G/A per 90, of 22.5. This is not a very useful visualization with this outlier, so we are going to revisualize this by filtering for players who have played at least 90 minutes in the tournament. This also serves as a reminder when we visualize our per 90 minute data, that we should be filtering for players who have played at least 90 minutes, because otherwise, we will get some very skewed data.

```
[ ]: # filter out players with 90 or more minutes played
     players_90min = players[players['minutes'] >= 90]

     plt.figure(figsize=(10, 6))
     sns.set_style('darkgrid')

     sns.barplot(x='goals_assists_per90', y='player', data=players_90min.
       ↪sort_values('goals_assists_per90', ascending=False).head(10), palette='Set2')
     plt.title('Top Goals + Assists (G/A) per 90 minutes in the World Cup',␣
       ↪fontsize=20)
     plt.suptitle('* players with at least 90 minutes played', fontsize=15)
     plt.xlabel('G/A', fontsize=15)
plt.ylabel('Player', fontsize=15)
```

```
[ ]: Text(0, 0.5, 'Player')
```



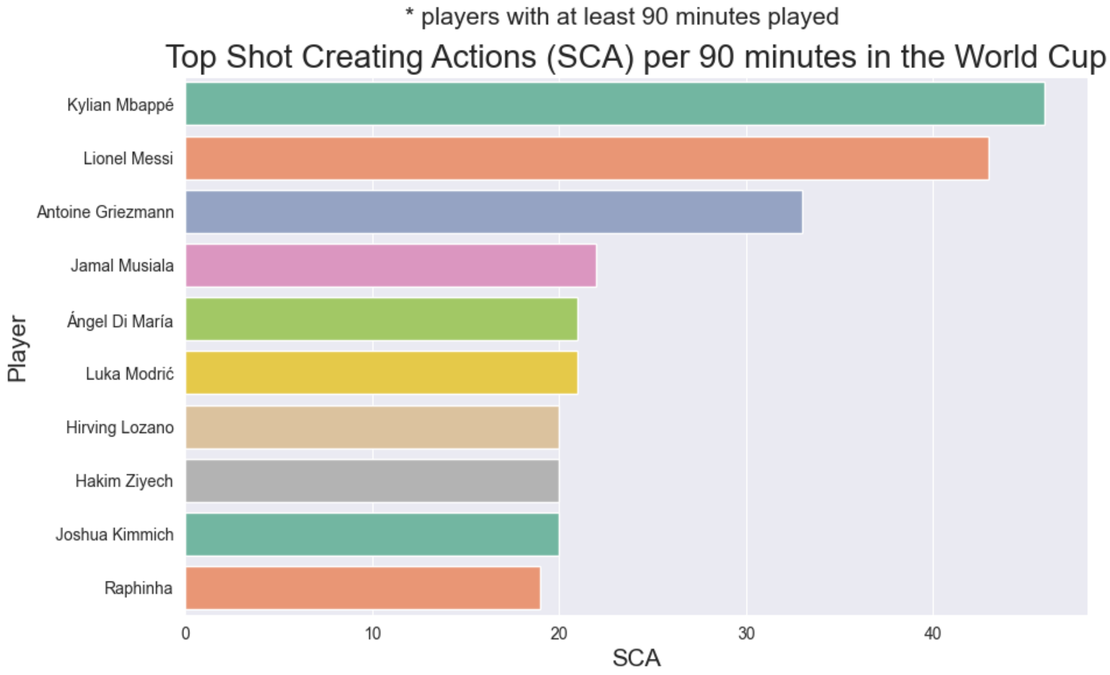
While Mbappe and Messi are still in the top 10 G/A per 90, some other players have emerged, and Goncalo Ramos (of Portugal) has actually taken the top spot. Him and many other players like Marcus Rashford (England) have emerged in this top 10, likely because they have played less minutes than the other players, but have still been able to produce at a high level when they are on the pitch.

Observing SCA and SCA per 90 SCA is an advanced statistic that is defined as a shot-creating action, which is a pass, dribble, or drawing of a foul that leads to a shot. This is an interesting statistic because it can show us who is creating the most chances for their team, and who is the most involved in the attack. SCA per 90 is a metric that is used to measure the total amount of shot-creating actions that a player has per 90 minutes played. We will be exploring these metrics to determine which players are deemed most efficient in the tournament based on these criteria.

```
[ ]: plt.figure(figsize=(10, 6))
     sns.set_style('darkgrid')

     sns.barplot(x='sca', y='player', data=players.sort_values('sca',␣
       ↪ascending=False).head(10), palette='Set2')
     plt.title('Top Shot Creating Actions (SCA) per 90 minutes in the World Cup',␣
       ↪fontsize=20)
```

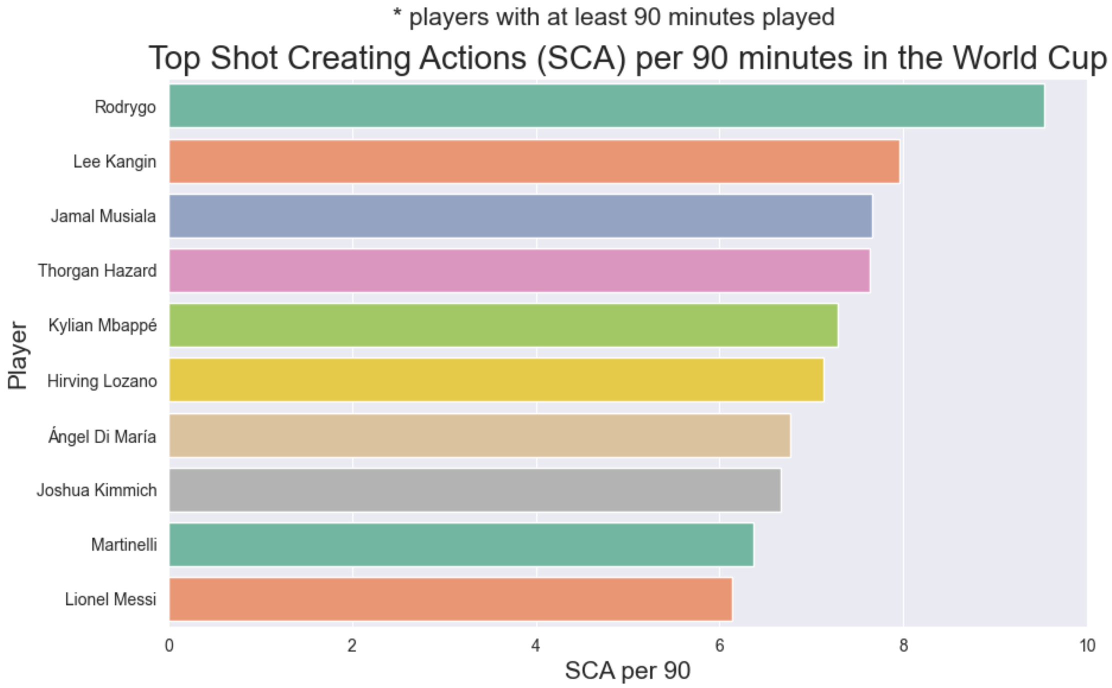
```
plt.suptitle('* players with at least 90 minutes played', fontsize=15)
plt.xlabel('SCA', fontsize=15)
plt.ylabel('Player', fontsize=15)
```

```
[ ]: Text(0, 0.5, 'Player')
```



```
[ ]: plt.figure(figsize=(10, 6))
     sns.set_style('darkgrid')

     sns.barplot(x='sca_per90', y='player', data=players_90min.
       ↪sort_values('sca_per90', ascending=False).head(10), palette='Set2')
     plt.title('Top Shot Creating Actions (SCA) per 90 minutes in the World Cup',␣
       ↪fontsize=20)
     plt.suptitle('* players with at least 90 minutes played', fontsize=15)
     plt.xlabel('SCA per 90', fontsize=15)
     plt.ylabel('Player', fontsize=15)
```

```
[ ]: Text(0, 0.5, 'Player')
```



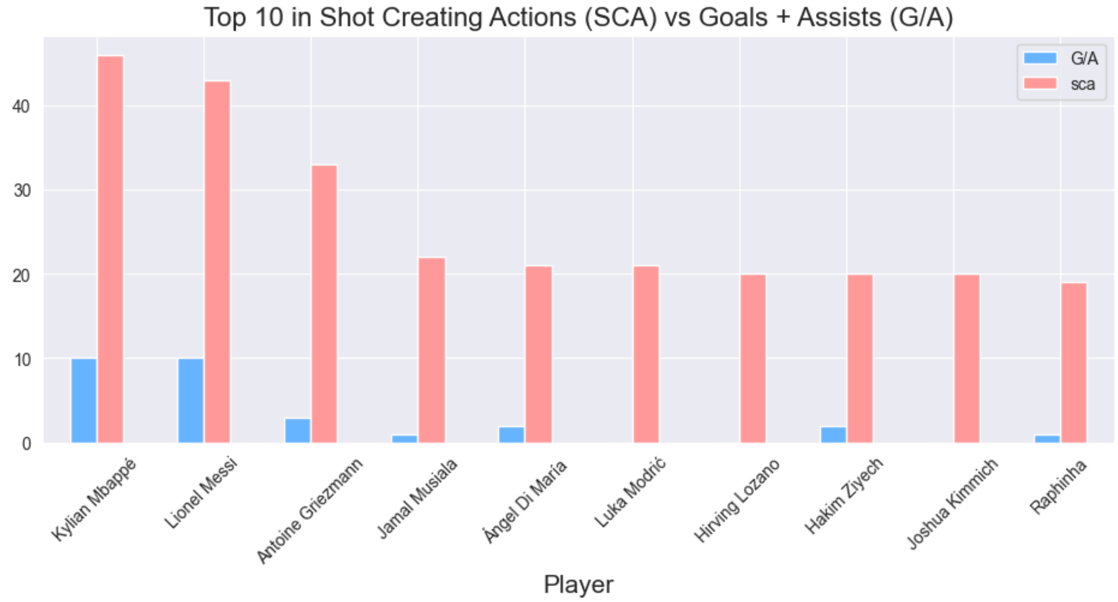
SCA is an advanced statistic that is defined as a shot-creating action, which is a pass, dribble, or drawing of a foul that leads to a shot. This is an interesting statistic because it can show us who is creating the most chances for their team, and who is the most involved in the attack. Here, we are seeing many new names, like Rodrygo and Gabriel Martinelli of Brazil, as well as some familar names like Lionel Messi and Kylian Mbappe.

Putting it all together Ultimately, we are curious how SCA actually coorelates with G/A, so we are going to visualize that relationship. In other words; our leaders in SCA, are they also our leaders in G/A?

[ ]:

```
_top_sca = players.sort_values('sca', ascending=False).head(10)
top_sca = _top_sca[['player', 'G/A', 'sca']]
top_sca = top_sca.set_index('player')


sns.set_style('darkgrid')
top_sca.plot.bar(figsize=(10, 6), color=['#66b3ff','#ff9999'])

plt.title('Top 10 in Shot Creating Actions (SCA) vs Goals + Assists (G/A)',␣
  ↪fontsize=16)
plt.xlabel('Player', fontsize=15)

plt.tight_layout(pad=3.0)
plt.xticks(rotation=45)
```

```
[ ]: (array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
      [Text(0, 0, 'Kylian Mbappé'),
       Text(1, 0, 'Lionel Messi'),
       Text(2, 0, 'Antoine Griezmann'),
       Text(3, 0, 'Jamal Musiala'),
       Text(4, 0, 'Ángel Di María'),
       Text(5, 0, 'Luka Modrić'),
       Text(6, 0, 'Hirving Lozano'),
       Text(7, 0, 'Hakim Ziyech'),
       Text(8, 0, 'Joshua Kimmich'),
       Text(9, 0, 'Raphinha')])
```



In total, there *does* seem to be a coorelation between SCA and G/A, but it is not necessarily a very strong one. While we do see some of the same names, like Mbappe, Messi, and the king of advanced assist statistics in Antoine Griezmann, we also see some new names, like Luka Modric, Hirving Lozano, and Joshua Kimmich, who have 0 G/A combined. This is an interesting observation, because it shows that while SCA is a good indicator of goal scoring opportunities (which in turn leads to goals and/or assists), it is not necessarily a strong indicator of G/A. However, it is still a good indicator of who is creating the most chances for their team, and who is the most involved in the attack.

```
[ ]: player_keepers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 41 entries, 0 to 40
Data columns (total 22 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
```

```
 0   player                   41 non-null     object
 1   club                     41 non-null     object
 2   position                 41 non-null     object
 3   age                      41 non-null     object
 4   team                     41 non-null     object
 5   birth_year               41 non-null     int64
 6   minutes_90s              41 non-null     float64
 7   goals_against            41 non-null     int64
 8   goals_against_per90      41 non-null     float64
 9   shots_on_target_against  41 non-null     int64
 10  saves                    41 non-null     int64
 11  save_pct                 40 non-null     float64
 12  wins                     41 non-null     int64
 13  ties                     41 non-null     int64
 14  losses                   41 non-null     int64
 15  clean_sheets             41 non-null     int64
 16  clean_sheets_pct         40 non-null     float64
 17  pens_att                 41 non-null     int64
 18  pens_allowed             41 non-null     int64
 19  pens_saved               41 non-null     int64
 20  pens_missed              41 non-null     int64
 21  pens_save_pct            17 non-null     float64
dtypes: float64(5), int64(12), object(5)
memory usage: 7.4+ KB
```

[ ]:

```
colors = ['#66b3ff','#ff9999','#99ff99','#ffcc99']
# create argentina_scorers, as all players from argentina that scored at least␣
 ↪one goal
argentina_scorers = players[(players['team'] == 'Argentina') &␣
 ↪(players['goals'] > 0)]

argentina_scorers
```

[ ]:

```
                  player             club position  age       team  \
30   Alexis Mac Allister         Brighton       MF   23  Argentina
187       Enzo Fernández          Benfica       MF   21  Argentina
330       Julián Álvarez  Manchester City       FW   22  Argentina
388         Lionel Messi        Paris S-G       FW   35  Argentina
470        Nahuel Molina  Atlético Madrid       DF   24  Argentina
675       Ángel Di María         Juventus       MF   34  Argentina

     birth_year  minutes_90s  tackles  tackles_won  tackles_def_3rd  ...  \
30         1998          6.1      9.0            4              3.0  ...
187        2001          6.3     22.0           15             15.0  ...
330        2000          5.2      4.0            2              3.0  ...
388        1987          7.7      5.0            3              1.0  ...
470        1998          6.3      8.0            4              6.0  ...
```

```
675        1988          3.2      3.0            1              2.0  …

     goals_per90  assists_per90  goals_assists_per90  minutes  \
30          0.16           0.16                 0.33    550.0
187         0.16           0.16                 0.32    564.0
330         0.78           0.00                 0.78    464.0
388         0.91           0.39                 1.30    690.0
470         0.16           0.16                 0.32    567.0
675         0.31           0.31                 0.62    288.0

     minutes_per_game   sca  sca_per90  gca  gca_per90  G/A
30               92.0  16.0       2.77  2.0       0.35    2
187              81.0  17.0       3.04  4.0       0.71    2
330              66.0  10.0       1.94  2.0       0.39    4
388              99.0  43.0       6.14  5.0       0.71   10
470              81.0  10.0       1.59  2.0       0.32    2
675              58.0  21.0       6.77  4.0       1.29    2

[6 rows x 92 columns]
```

Another race to follow during the World Cup is that of the Best Young Player award, which is given to the best player in the tournament under the age of 21. Many factors contribute to the award, including how far the player's team goes in the tournament, and how well they perform. We are going to look at some of the stats of the top 5 players under 21 in the tournament, and see how they compare to each other, and ultimately who should have won the award based on statistics.

1.5 Visualizing the rounds of the tournament

There is an infinite amount of categories that can be observed, and a long list of interesting trends, features, statistics, and more than can be analyzed for each match, and for each round. Our research question in specifics to each round is: *As the rounds go on, do teams tend to play more conservatively, or do they play more aggressively?* To answer this question, there are a few things we want to look at generally for each round as a whole. It should be noted that if there is an interest at observing a specific match, I have created a tool that allows you to select a match, and observe the statistics for that match. You can find that tool here.

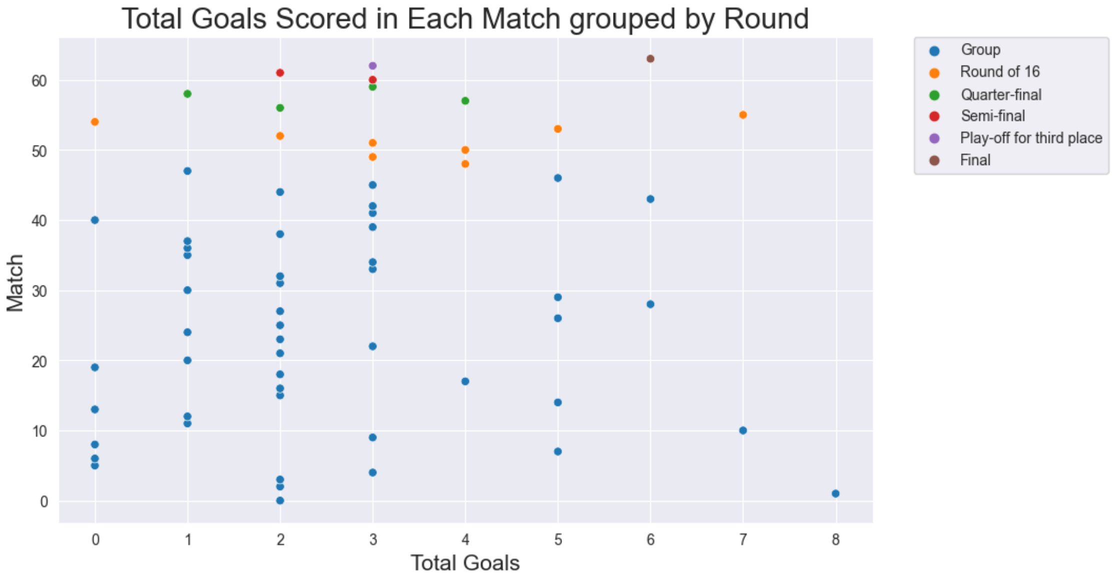
The features we will be comparing for each round are:

- Total goals scored in each match of the round
- Total number of passes in each match of the round
- Total amount of fouls in each match of the round

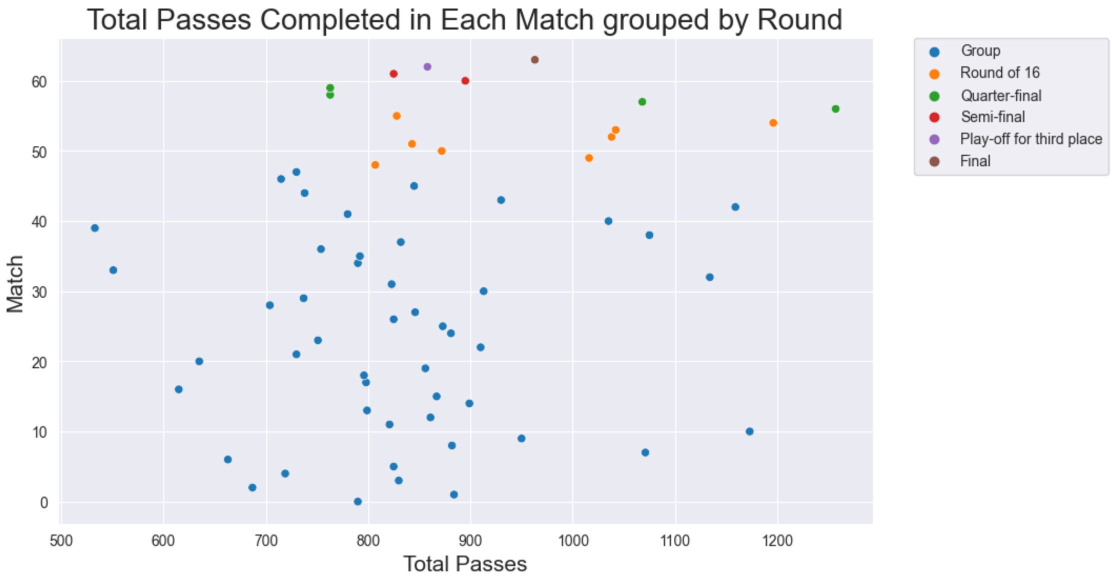
[ ]:

```
matches['total_goals'] = matches['number of goals team1'] + matches['number of␣
↪goals team2']

plt.figure(figsize=(10, 6))
sns.set_style('darkgrid')
sns.scatterplot(data=matches, y=matches.index, x='total_goals', hue='category')
```

```
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.title('Total Goals Scored in Each Match grouped by Round', fontsize=20)
plt.xlabel('Total Goals', fontsize=15)
plt.ylabel('Match', fontsize=15)
plt.show()
# different kind of plot??
```



[ ]:
```
matches['total_passes'] = matches['passes completed team1'] + matches['passes␣
 ↪completed team2']

plt.figure(figsize=(10, 6))
sns.set_style('darkgrid')
sns.scatterplot(data=matches, y=matches.index, x='total_passes', hue='category')

plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.title('Total Passes Completed in Each Match grouped by Round', fontsize=20)
plt.xlabel('Total Passes', fontsize=15)
plt.ylabel('Match', fontsize=15)
plt.show()
# different kind of plot??
```



A trend we're able to see here is that a majority of matches throughout the tournament end up with around 800-900 total passes completed by both teams combined. It is interesting to note that outside of 900 passes, it seems to be an even split between group play matches and elimination matches.

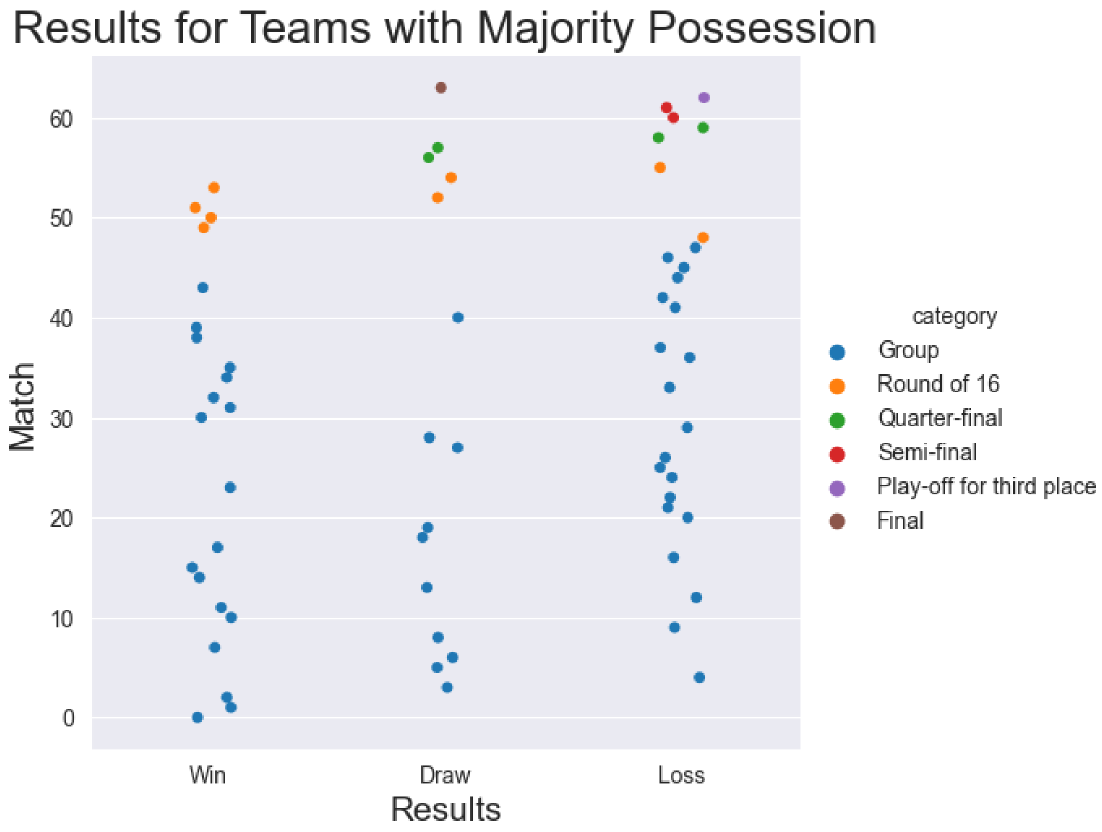
```
[ ]: matches['possession win'] = ''
     possession_wins = 0
     possession_losses = 0
     for i, row in matches.iterrows():
         if row['number of goals team1'] != row['number of goals team2']:
             if (row['possession team1'] > row['possession team2'] and row['number␣
       ↪of goals team1'] > row['number of goals team2']) or (row['possession team2']␣
       ↪> row['possession team1'] and row['number of goals team2'] > row['number of␣
       ↪goals team1']):
                 matches.at[i, 'possession win'] = 'Win'
                 possession_wins += 1
             else:
                 possession_losses += 1
                 matches.at[i, 'possession win'] = 'Loss'
         else:
             matches.at[i, 'possession win'] = 'Draw'

     print(f'Teams with more possession won {possession_wins} games, and lost␣
       ↪{possession_losses} games.')
```

```
Teams with more possession won 22 games, and lost 27 games.
```

[ ]:

```
plt.figure(figsize=(10, 6))
sns.set_style('darkgrid')
sns.catplot(data=matches, y=matches.index, x='possession win', hue='category')

plt.title('Results for Teams with Majority Possession', fontsize=20)
plt.xlabel('Results', fontsize=15)
plt.ylabel('Match', fontsize=15)
plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```



We can see that broken down by round:

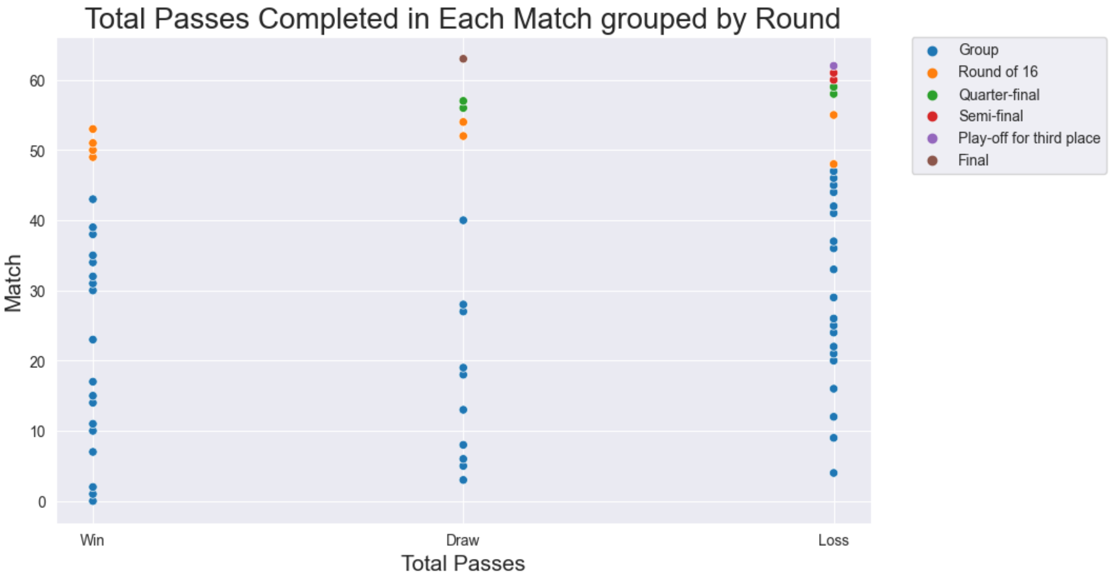
- More teams lost with majority possession in group play
- The round of 16 was very close, and the distribution of wins was very even
- A similar trend can be seen in the quarterfinals, but it can be noted that no team with majority possession won in the quarterfinals
- No team in the semifinals won with possession
- The final resulted in a draw after extra time, and went to penalty kicks

- The third place match resulted in a loss for the team that held majority possession

So, we can see that possession is not a very good indicator of success in the World Cup, and that it is not a very good indicator of success in the later rounds of the tournament. This is interesting because it is a common belief that the team that holds majority possession is the team that is more likely to win the match.

[ ]:

```
plt.figure(figsize=(10, 6))
sns.set_style('darkgrid')
sns.scatterplot(data=matches, y=matches.index, x='possession win',␣
 ↪hue='category')

plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.title('Total Passes Completed in Each Match grouped by Round', fontsize=20)
plt.xlabel('Total Passes', fontsize=15)
plt.ylabel('Match', fontsize=15)
plt.show()
```

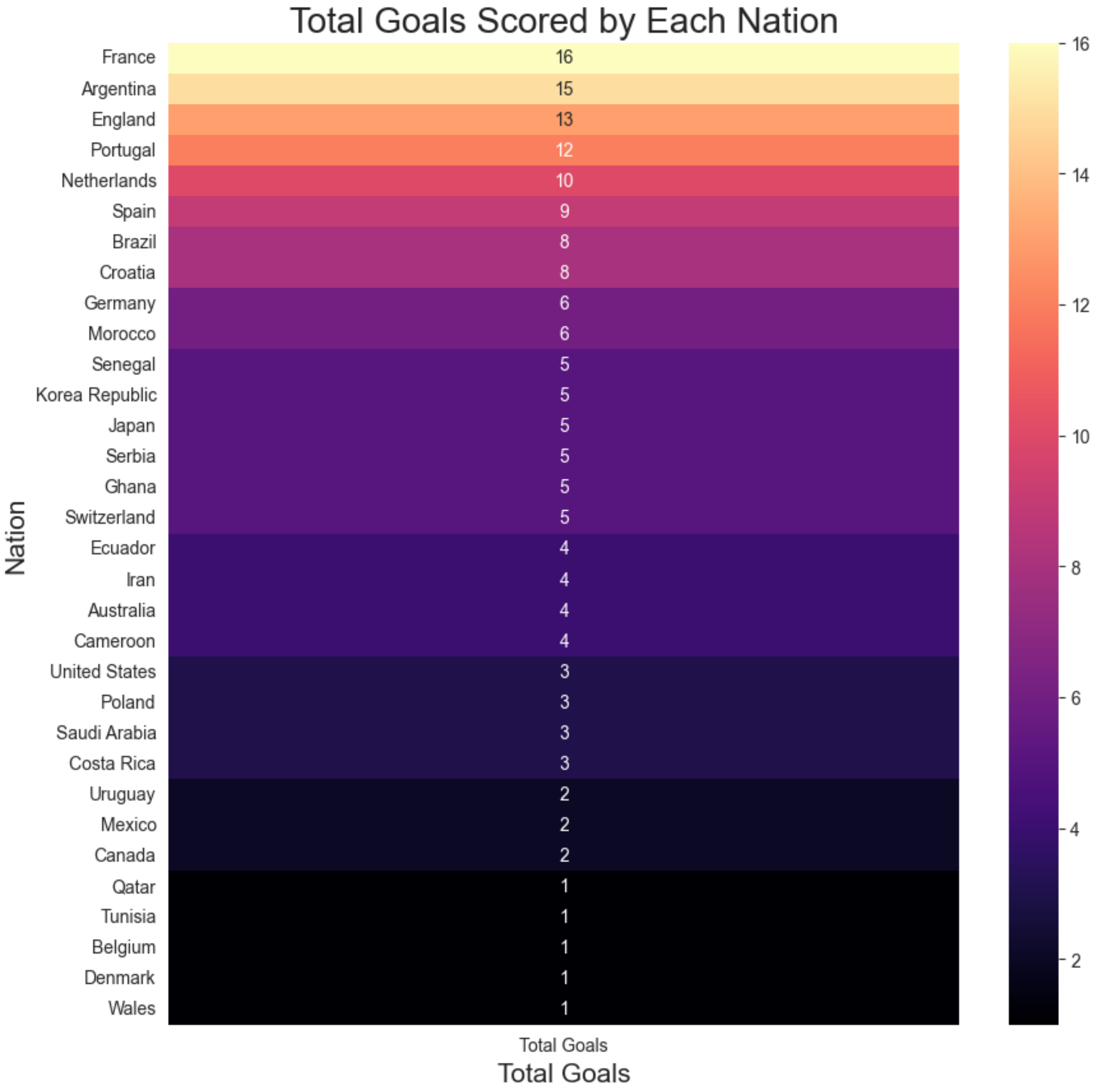


This goes to show that the concept of having more possession does not necessarily mean that you will win the match, as we can see that the teams that had more possession actually lost the match more often than not. This is interesting because it goes against the common belief that having more possession means that you will win the match, and that in high pressure tournament situations, there is room for many strategies to win. For example, in the knockout match between the United States and the Netherlands, although the United States had majority of possession, the Netherlands played to their tactics and were in complete control of the game, as they were able to win 3-1.

1.5.1 Visualizing the Group Stage

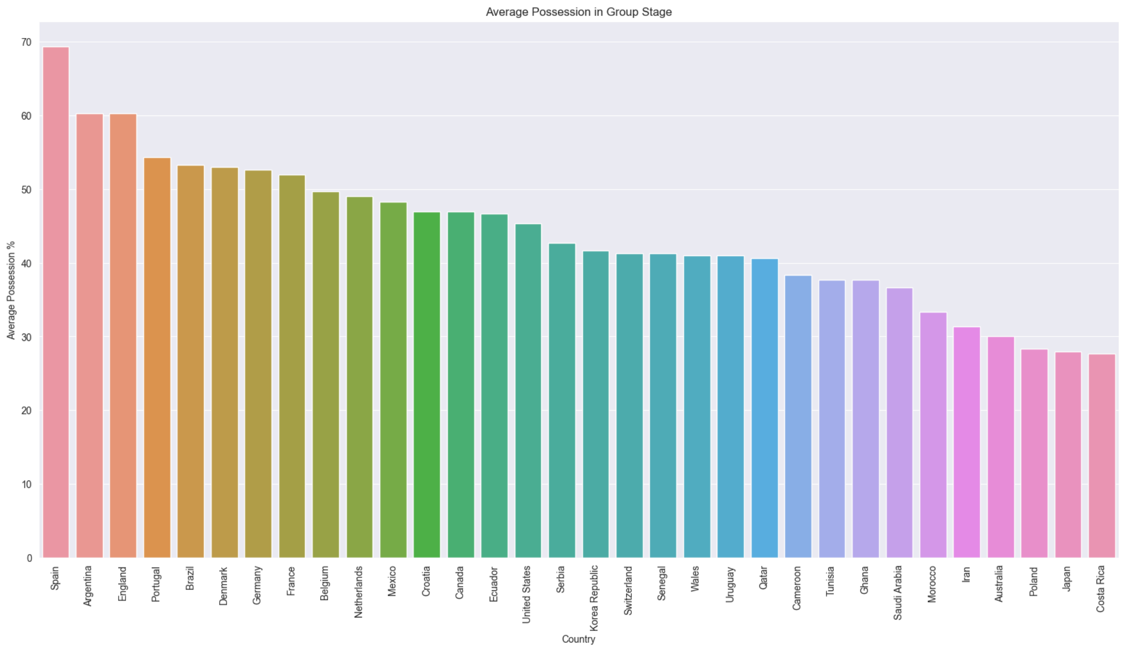
[ ]:

```
# create a heatmap using the Total Goals column
plt.figure(figsize=(10, 10))
sns.set_style('darkgrid')

sns.heatmap(countries[['Total Goals']].sort_values(by='Total Goals', ascending=False), annot=True, cmap='magma')
plt.title('Total Goals Scored by Each Nation', fontsize=20)
plt.xlabel('Total Goals', fontsize=15)
plt.ylabel('Nation', fontsize=15)
plt.show()
```

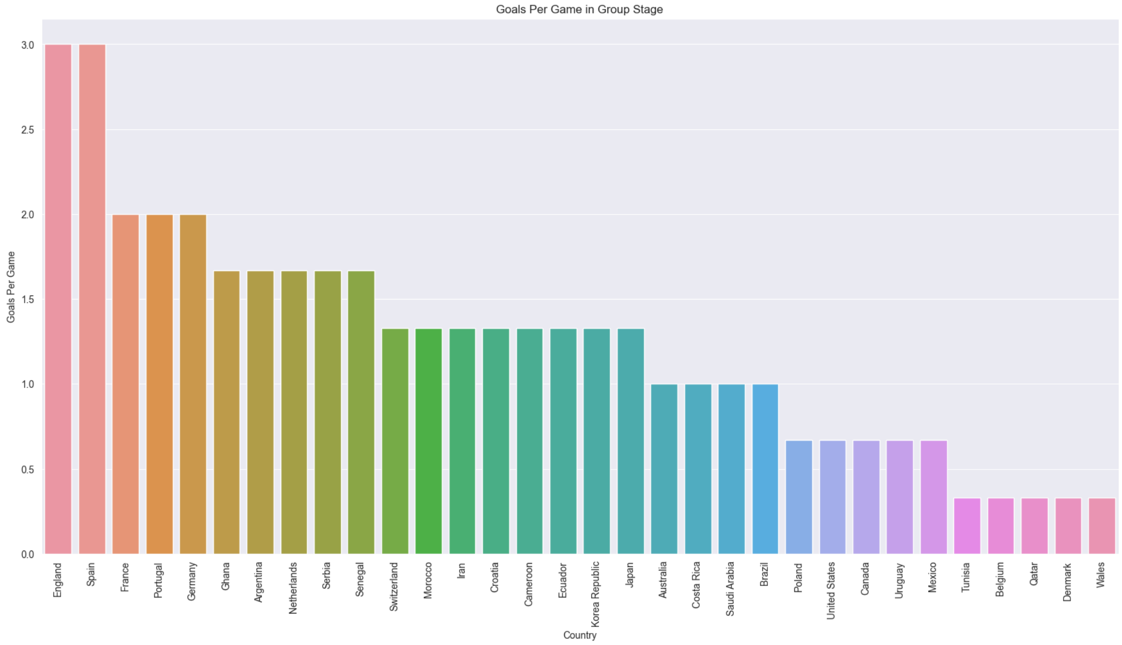


It makes sense that France and Argentina were the two top scorers tournament wide, because they were the two finalists, and went all the way.

[ ]:
```
# i want this to be colored by groupid but for some reason it doesnt want to␣
 ↪work...any ideas?
plt.figure(figsize=(20, 10))
sns.set_style('darkgrid')

sns.barplot(x=countries['Average Possession in Group'].
 ↪sort_values(ascending=False).index, y=countries['Average Possession in␣
 ↪Group'].sort_values(ascending=False))
plt.xticks(rotation=90)
plt.xlabel('Country')
plt.ylabel('Average Possession %')
plt.title('Average Possession in Group Stage')
plt.show()
```

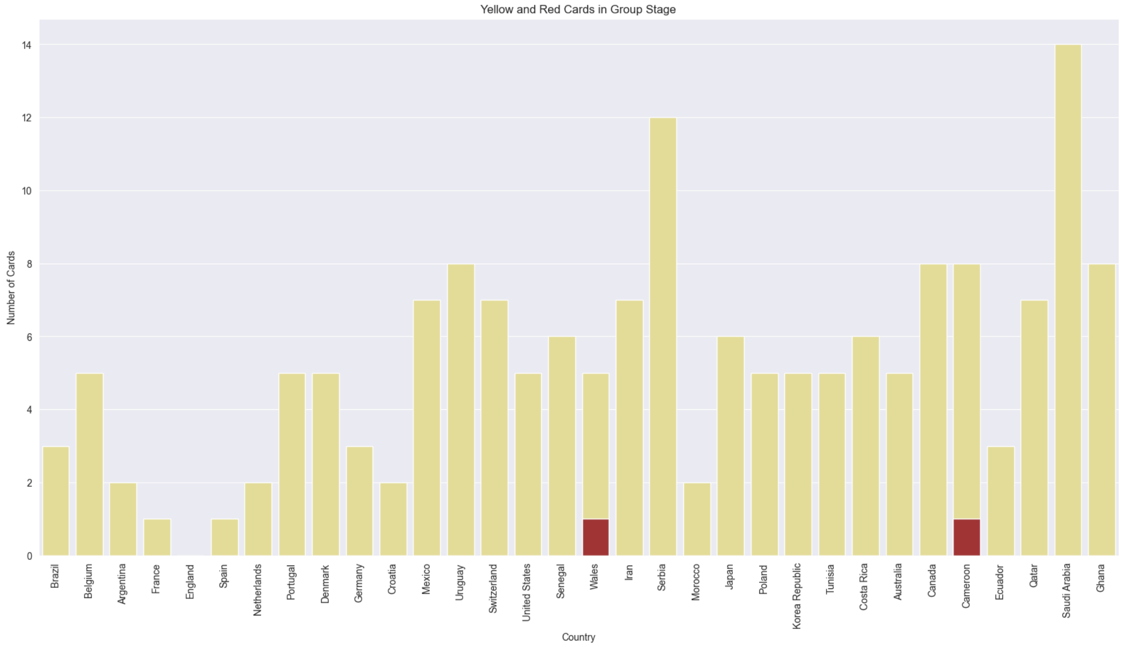


[ ]:
```
plt.figure(figsize=(20, 10))
sns.set_style('darkgrid')

sns.barplot(x=countries['Goals Per Game in Group'].sort_values(ascending=False).
 ↪index, y=countries['Goals Per Game in Group'].sort_values(ascending=False))
plt.xticks(rotation=90)
plt.xlabel('Country')
plt.ylabel('Goals Per Game')
plt.title('Goals Per Game in Group Stage')
plt.show()
```



It appears that this particular visualization gives us many different groups in tiers in which they found goals in the group stages. England and Spain were king among them with 3 goals averaged per match (who were also both in the top 3 of possession as well), while the likes of Tunisia, Belgium, Qatar, Denmark, and Wales were among the lowest with under 0.5 goals per match. Let's look further at these two kings of the Group Stage to see what else they have in common.

```
[ ]: # create a multibar bar graph comparing yellow and red cards
     plt.figure(figsize=(20, 10))
     sns.set_style('darkgrid')

     sns.barplot(x=countries.index, y=countries['Yellow Cards in Group'],␣
      ↪color='khaki')
     sns.barplot(x=countries.index, y=countries['Red Cards in Group'],␣
      ↪color='firebrick')
     plt.xticks(rotation=90)
     plt.xlabel('Country')
     plt.ylabel('Number of Cards')
     plt.title('Yellow and Red Cards in Group Stage')
     plt.show()
```



We can see that there were a lot of yellow cards distributed during the group stage, but not as many red cards. Wales and Cameroon were the only nations to have a player get sent off during the group stages, and England was the only nation not to pick up a single card during the round. Saudia Arabia, followed by Serbia were by far the most carded nations.

1.5.2 Visualizing the Round of 16

1.5.3 Visualizing the Quarter-Finals

1.5.4 Visualizing the Semi-Finals

1.5.5 Visualizing the Third-Place Match

1.5.6 Visualizing the Final