FIFA World Cup Performance Analysis

```
import numpy as np
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from plotly.figure_factory import create_distplot
import re
from tabulate import tabulate
import plotly.io as pio
```

```
pio.templates.default = 'seaborn'
```

Load Data

```
fifa_ranking = pd.read_csv('fifa_ranking_2022-10-06.csv')
matches = pd.read_csv('matches_1930_2022.csv')
world_cup = pd.read_csv('world_cup.csv')
```

```
print(fifa_ranking.shape,fifa_ranking.columns)
fifa_ranking.head()
```

```
(211, 7) Index(['team', 'team_code', 'association', 'rank', 'previous_rank', 'points',
       'previous_points'],
      dtype='object')
```

	team	team_code	association	rank	previous_rank	points	previous_points
0	Brazil	BRA	CONMEBOL	1	1	1841.30	1837.56
1	Belgium	BEL	UEFA	2	2	1816.71	1821.92
2	Argentina	ARG	CONMEBOL	3	3	1773.88	1770.65
3	France	FRA	UEFA	4	4	1759.78	1764.85
4	England	ENG	UEFA	5	5	1728.47	1737.46

Next steps: Generate code with `fifa_ranking` | View recommended plots | New interactive sheet

```
print(matches.shape,matches.columns)
matches.head()
```

```
(964, 32) Index(['home_team', 'away_team', 'home_score', 'home_xg', 'home_penalty',
       'away_score', 'away_xg', 'away_penalty', 'home_manager', 'home_captain',
       'away_manager', 'away_captain', 'Attendance', 'Venue', 'Officials',
       'Round', 'Date', 'Score', 'Referee', 'Notes', 'Host', 'Year',
       'home_goal', 'away_goal', 'home_goal_long', 'away_goal_long',
       'home_own_goal', 'away_own_goal', 'home_penalty_goal',
       'away_penalty_goal', 'home_substitute_in_long',
       'away_substitute_in_long'],
      dtype='object')
```

	home_team	away_team	home_score	home_xg	home_penalty	away_score	away_xg	away_penalty	home_manager	home_captain	...	home_goal	away_goal	home_goal_l
0	Argentina	France	3	3.3	4.0	3	2.2	2.0	Lionel Scaloni	Lionel Messi	...	Ángel Di María · 36|Lionel Messi · 108	Kylian Mbappé · 81	['36’|2:0|Ár Di María|Assist:|Alex
1	Croatia	Morocco	2	0.7	NaN	1	1.2	NaN	Zlatko Dalić	Luka Modrić	...	Joško Gvardiol · 7|Mislav Oršić · 42	Achraf Dari · 9	['7’|1:0|Jo Gvardiol|Assist:|I F
2	France	Morocco	2	2.0	NaN	0	0.9	NaN	Didier Deschamps	Hugo Lloris	...	Theo Hernández · 5|Randal Kolo Muani · 79	NaN	['5’|1:0|Th Hernánd '79’
3	Argentina	Croatia	3	2.3	NaN	0	0.5	NaN	Lionel Scaloni	Lionel Messi	...	Julián Álvarez · 39|Julián Álvarez · 69	NaN	['39’|2:0|Ju Álvarez', '69&rsquo
4	Morocco	Portugal	1	1.4	NaN	0	0.9	NaN	Hoalid Regragui	Romain Saïss	...	Youssef En-Nesyri · 42	NaN	['42’|1:0|Yous En-Nesyri|Assist:|Ya

5 rows × 32 columns

```
print(world_cup.shape,world_cup.columns)
world_cup.head()
```

```
(22, 9) Index(['Year', 'Host', 'Teams', 'Champion', 'Runner-Up', 'TopScorrer',
       'Attendance', 'AttendanceAvg', 'Matches'],
      dtype='object')
```

	Year	Host	Teams	Champion	Runner-Up	TopScorrer	Attendance	AttendanceAvg	Matches
0	2022	Qatar	32	Argentina	France	Kylian Mbappé - 8	3404252	53191	64
1	2018	Russia	32	France	Croatia	Harry Kane - 6	3031768	47371	64
2	2014	Brazil	32	Germany	Argentina	James Rodríguez - 6	3429873	53592	64
3	2010	South Africa	32	Spain	Netherlands	Wesley Sneijder, Thomas Müller... - 5	3178856	49670	64
4	2006	Germany	32	Italy	France	Miroslav Klose - 5	3352605	52384	64

Next steps: Generate code with world_cup | View recommended plots | New interactive sheet

Null Values

```
fifa_ranking.isnull().sum()/len(fifa_ranking)*100
```

	0
team	0.0
team_code	0.0
association	0.0
rank	0.0
previous_rank	0.0
points	0.0
previous_points	0.0

dtype: float64

```
world_cup.isnull().sum()/len(world_cup)*100
```

	0
Year	0.0
Host	0.0
Teams	0.0
Champion	0.0
Runner-Up	0.0
TopScorrer	0.0
Attendance	0.0
AttendanceAvg	0.0
Matches	0.0

dtype: float64

```
matches.isnull().sum()/len(matches)*100
```

	0
home_team	0.000000
away_team	0.000000
home_score	0.000000
home_xg	86.721992
home_penalty	96.369295
away_score	0.000000
away_xg	86.721992
away_penalty	96.369295
home_manager	0.000000
home_captain	33.195021
away_manager	0.000000
away_captain	33.195021
Attendance	0.000000
Venue	0.000000
Officials	26.452282
Round	0.000000
Date	0.000000
Score	0.000000
Referee	26.452282
Notes	92.427386
Host	0.000000
Year	0.000000
home_goal	25.518672
away_goal	40.767635
home_goal_long	25.518672
away_goal_long	40.767635
home_own_goal	95.954357
away_own_goal	98.236515
home_penalty_goal	87.966805

home_penalty_goal	87.900803
away_penalty_goal	91.286307
home_substitute_in_long	23.236515
away_substitute_in_long	22.510373

dtype: float64

```
print(fifa_ranking.duplicated().sum())
print(matches.duplicated().sum())
print(world_cup.duplicated().sum())
```

```
0
0
0
```

```
matches['home_team'].unique()
```

```
array(['Argentina', 'Croatia', 'France', 'Morocco', 'England',
       'Netherlands', 'Portugal', 'Japan', 'Brazil', 'Korea Republic',
       'Ghana', 'Cameroon', 'Serbia', 'Canada', 'Costa Rica', 'Australia',
       'Tunisia', 'Saudi Arabia', 'Poland', 'Ecuador', 'IR Iran', 'Wales',
       'Belgium', 'Spain', 'Qatar', 'Switzerland', 'Uruguay', 'Germany',
       'Denmark', 'Mexico', 'Senegal', 'United States', 'Sweden',
       'Russia', 'Colombia', 'Panama', 'Iceland', 'Nigeria', 'Peru',
       'Egypt', 'Algeria', 'Bosnia and Herzegovina', 'Honduras', 'Italy',
       'Greece', "Côte d'Ivoire", 'Chile', 'Paraguay', 'Korea DPR',
       'Slovakia', 'Slovenia', 'South Africa', 'New Zealand', 'Ukraine',
       'Togo', 'Czech Republic', 'Serbia and Montenegro', 'Angola',
       'Trinidad and Tobago', 'Türkiye', 'China PR',
       'Republic of Ireland', 'Romania', 'Scotland', 'FR Yugoslavia',
       'Jamaica', 'Bulgaria', 'Bolivia', 'Norway', 'West Germany',
       'Yugoslavia', 'Czechoslovakia', 'Austria', 'United Arab Emirates',
       'Soviet Union', 'Northern Ireland', 'Iraq', 'Hungary',
       'Germany DR', 'Zaire', 'Haiti', 'Cuba'], dtype=object)
```

Entries for Germany

- **Germany**
- **West Germany**
- **German DR (East Germany)**

Historical Context

From 1949 to 1990, Germany was divided into two separate states:

1. **West Germany**
2. **German DR (East Germany)**

In 1990, both West Germany and East Germany merged to form the unified state known as **Germany**.

```
matches['home_team'] = matches['home_team'].apply(lambda x: x.replace('Germany DR', 'West Germany'))
matches['away_team'] = matches['away_team'].apply(lambda x: x.replace('Germany DR', 'West Germany'))
```

Total Team Scores

```
matches['home_penalty'].fillna(0, inplace=True)
matches['away_penalty'].fillna(0, inplace=True)

matches['home_total'] = matches['home_score'] + matches['home_penalty']
matches['away_total'] = matches['away_score'] + matches['away_penalty']
```

```
def winner(row):
    if row['home_total']>row['away_total']:
        return row['home_team']
    elif row['home_total']<row['away_total']:
        return row['away_team']
    return 'Draw'
matches['winner'] = matches.apply(winner,axis=1)
```

```
knockouts = ['Final', 'Semi-finals', 'Quarter-finals']
matches['knockout'] = matches['Round'].apply(lambda x: 'Knockout' if x in knockouts else 'Non-Knockout')
```

```
print(f'Total number of matches: {len(matches)}')
print(f"Total goals scored: {sum(matches['home_total'] + matches['away_total'])}")
print(f"Average goals scored: {sum(matches['home_total'] + matches['away_total'])/len(matches)}")
print(f"Total Attendance Over time: {sum(matches['Attendance'])}")
print(f"Average Attendance Over time: {sum(matches['Attendance'])/len(matches)}")
```

```
Total number of matches: 964
Total goals scored: 2942.0
Average goals scored: 3.0518672199170123
Total Attendance Over time: 44048413
Average Attendance Over time: 45693.3744813278
```

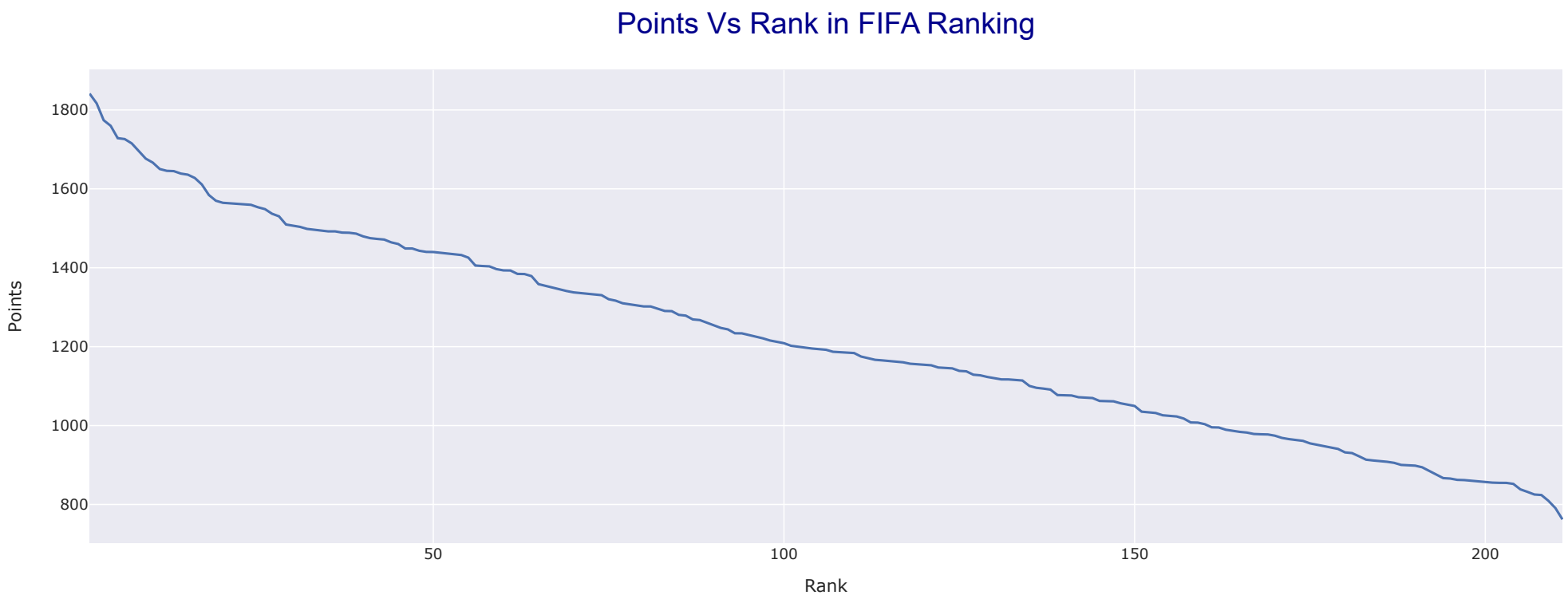
FIFA Ranking

```
fifa_ranking.columns
```

```
Index(['team', 'team_code', 'association', 'rank', 'previous_rank', 'points',
       'previous_points'],
      dtype='object')
```

Points Vs Ranking

```
fig = px.line(data_frame=fifa_ranking, x='rank', y='points',hover_name='team')
fig.update_layout(
    xaxis_title='Rank',
    yaxis_title='Points',
    title='Points Vs Rank in FIFA Ranking',
    title_x=0.5,
    title_font=dict(size=24, family='Arial, sans-serif', color='darkblue')
)
fig.show()
```



Finding whether ranking of a team is improved or not

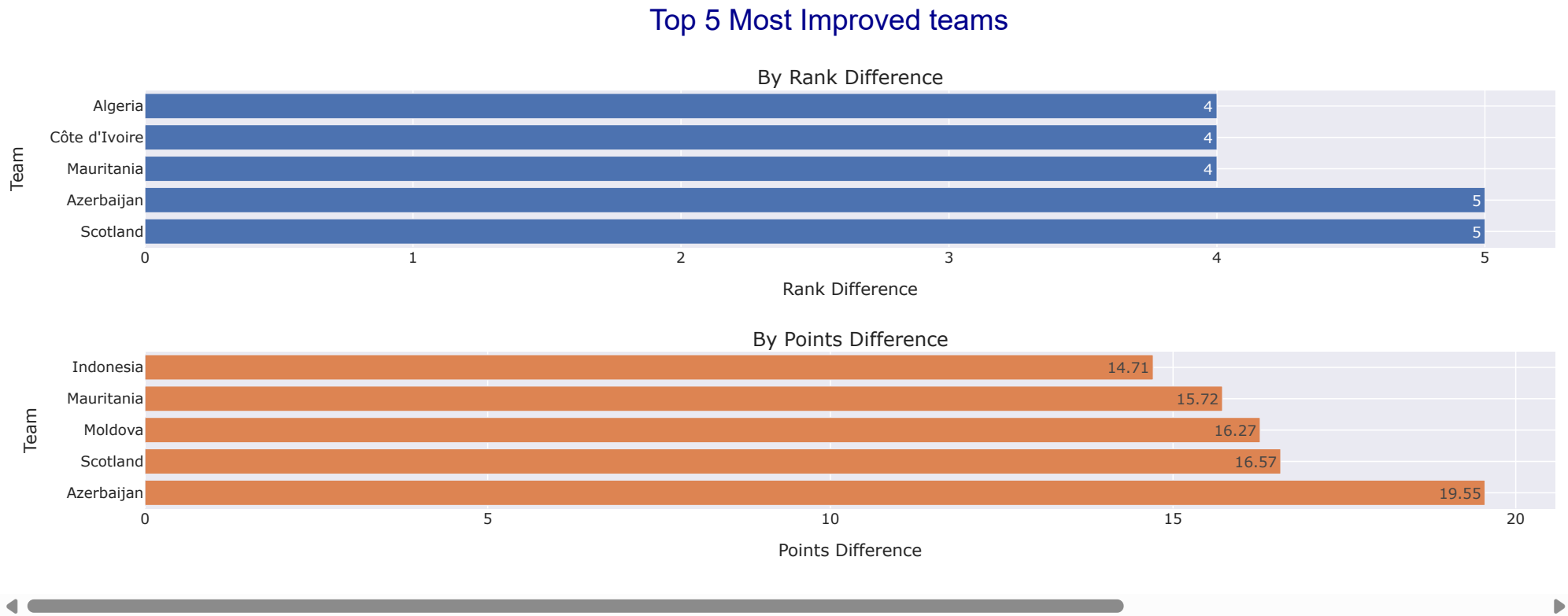
```
def rank_change(row):
    if row['rank'] == row['previous_rank']:
        return 'Same Rank'
    elif row['rank'] < row['previous_rank']:
        return 'Improved'
    else:
        return 'Declined'

fifa_ranking['rank_change'] = fifa_ranking.apply(rank_change, axis=1)
```

```
def change_in_rank(team):
    return fifa_ranking[fifa_ranking['team'] == team]['rank_change'].values

print(change_in_rank('Canada'))
```

```
['Improved']
```

Most Improved teams

```
fifa_ranking['rank_difference'] = fifa_ranking['previous_rank'] - fifa_ranking['rank']
fifa_ranking['points_difference'] = fifa_ranking['points'] - fifa_ranking['previous_points']

top_teams_rank = fifa_ranking[fifa_ranking['rank_change']=='Improved'].sort_values('rank_difference', ascending=False).head()
top_teams_points = fifa_ranking[fifa_ranking['rank_change']=='Improved'].sort_values('points_difference', ascending=False).head()

fig = make_subplots(rows=2, cols=1, subplot_titles=(
    'By Rank Difference',
    'By Points Difference'
))

rank = go.Bar(
        x=top_teams_rank['rank_difference'],
        y=top_teams_rank['team'],
        text=top_teams_rank['rank_difference'],
        name='Rank Difference',
        orientation='h'
    )

points = go.Bar(
        x=top_teams_points['points_difference'],
        y=top_teams_points['team'],
        text=np.round(top_teams_points['points_difference'],2),
        name='Points Difference',
        orientation='h'
    )

fig.add_trace(rank, row=1, col=1)
fig.add_trace(points, row=2, col=1)

fig.update_layout(
    title_text='Top 5 Most Improved teams',
    title_x=0.5,
    title_font=dict(size=24, family='Arial, sans-serif', color='darkblue'),
    xaxis_title='Rank Difference',
    yaxis_title='Team',
    xaxis2_title='Points Difference',
    yaxis2_title='Team',
    showlegend=False
)

fig.show()
```



Top Teams by Region/association

```
top_teams_by_region = fifa_ranking.groupby('association').apply(lambda x: x.sort_values('rank').head(1)).reset_index(drop=True)

print("Top Teams by Association/region:")
print(tabulate(top_teams_by_region[['association', 'team', 'rank']], headers='keys', tablefmt='pretty'))
```

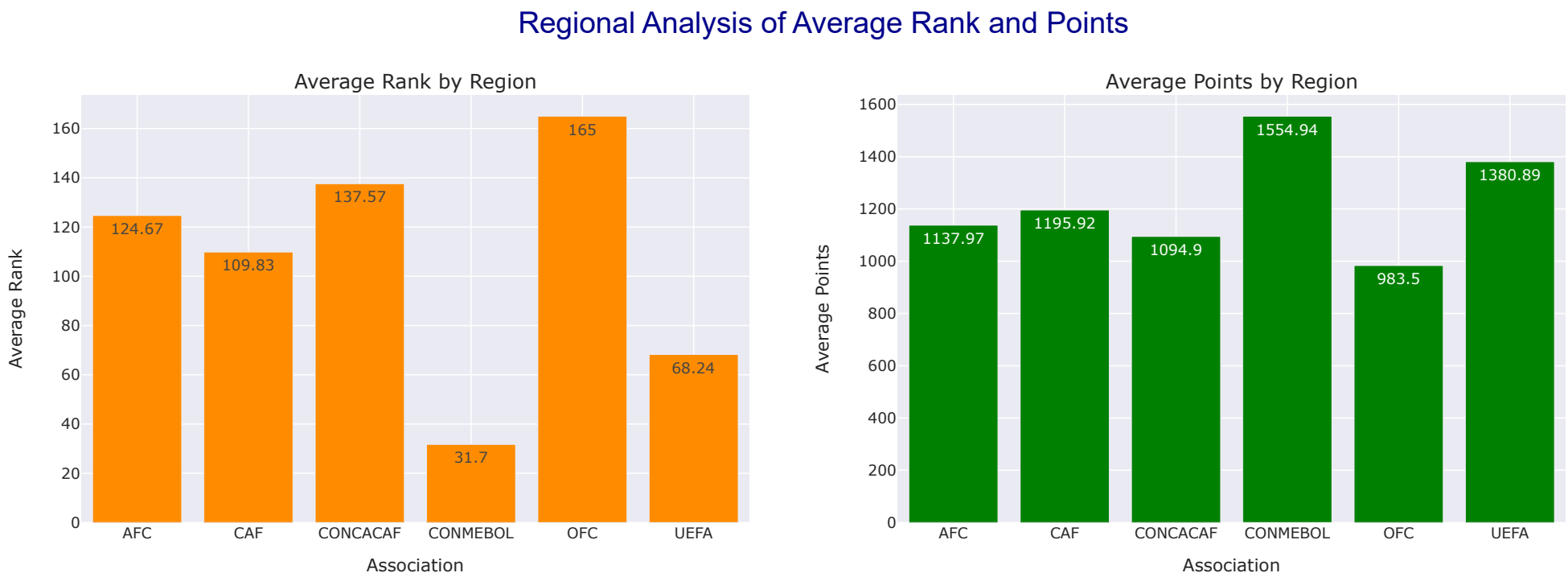
```
Top Teams by Association/region:
+---+-------------+-------------+------+
|   | association |    team     | rank |
+---+-------------+-------------+------+
| 0 |     AFC     |   IR Iran   |  20  |
| 1 |     CAF     |   Senegal   |  18  |
| 2 |  CONCACAF   |   Mexico    |  13  |
| 3 |  CONMEBOL   |   Brazil    |  1   |
| 4 |     OFC     | New Zealand | 105  |
| 5 |    UEFA     |   Belgium   |  2   |
+---+-------------+-------------+------+
```

Average Rank and Points in Each Association

```
fifa_ranking.groupby('association').agg({
    'rank': 'mean',
    'points': 'mean'
}).reset_index()
```

	association	rank	points
0	AFC	124.673913	1137.970000
1	CAF	109.833333	1195.924815
2	CONCACAF	137.571429	1094.896286
3	CONMEBOL	31.700000	1554.936000
4	OFC	165.000000	983.504545
5	UEFA	68.236364	1380.894364

```
region_stats = fifa_ranking.groupby('association').agg({
    'rank': 'mean',
    'points': 'mean'
}).reset_index()

rank = go.Bar(
        x=region_stats['association'],
        y=region_stats['rank'],
        name='Average Rank',
        text=np.round(region_stats['rank'],2),
        marker_color='darkorange'
    )
points = go.Bar(
        x=region_stats['association'],
        y=region_stats['points'],
        name='Average Points',
        text=np.round(region_stats['points'],2),
        marker_color='green'
    )

fig = make_subplots(rows=1, cols=2, subplot_titles=(
    'Average Rank by Region',
    'Average Points by Region'
))

fig.add_trace(rank, row=1, col=1)
fig.add_trace(points, row=1, col=2)

fig.update_layout(
    title_text='Regional Analysis of Average Rank and Points',
    title_x=0.5,
    title_font=dict(size=24, family='Arial, sans-serif', color='darkblue'),
    xaxis_title='Association',
    yaxis_title='Average Rank',
    xaxis2_title='Association',
    yaxis2_title='Average Points',
    showlegend=False
)

fig.show()
```



World Cup

```
world_cup.head()
```

	Year	Host	Teams	Champion	Runner-Up	TopScorrer	Attendance	AttendanceAvg	Matches
0	2022	Qatar	32	Argentina	France	Kylian Mbappé - 8	3404252	53191	64
1	2018	Russia	32	France	Croatia	Harry Kane - 6	3031768	47371	64
2	2014	Brazil	32	Germany	Argentina	James Rodríguez - 6	3429873	53592	64
3	2010	South Africa	32	Spain	Netherlands	Wesley Sneijder, Thomas Müller... - 5	3178856	49670	64
4	2006	Germany	32	Italy	France	Miroslav Klose - 5	3352605	52384	64

Next steps: Generate code with world_cup | View recommended plots | New interactive sheet

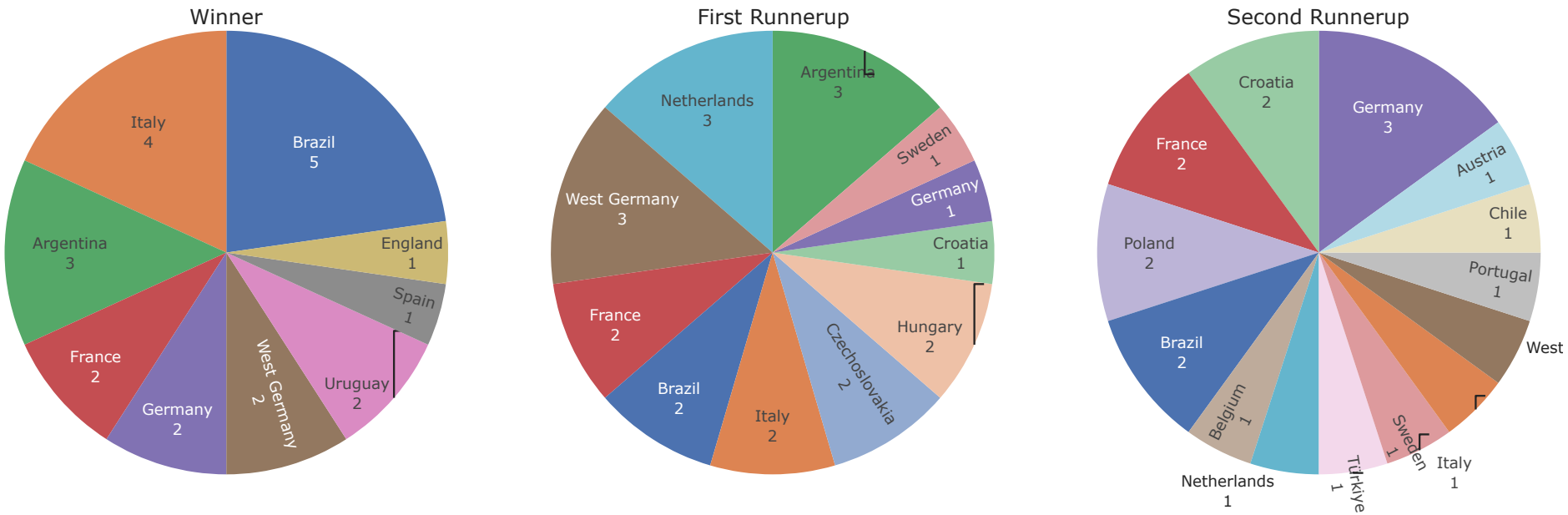
World Cup Attendance Over the Years

```
fig = px.line(data_frame=world_cup, x='Year', y='Attendance')
fig.update_layout(
    xaxis_title='Year',
    yaxis_title='Attendance (in Million)',
    title='World Cup Attendance Over the Years',
    title_x=0.5,
    title_font=dict(size=24, family='Arial, sans-serif', color='darkblue'),
    xaxis=dict(tickvals=world_cup['Year'].unique())
)
fig.show()
```

Champions and Runner-up Analysis

```
winner = world_cup['Champion'].value_counts().reset_index()
runner = world_cup['Runner-Up'].value_counts().reset_index()

def third_place(row):
    if row['home_total']>row['away_total']:
        return row['home_team']
    return row['away_team']
third = matches[matches['Round']=='Third-place match']
third['third_place'] = third.apply(third_place, axis=1)
third = third['third_place'].value_counts().reset_index()
```

```
trace1 = go.Pie(
    labels=winner['Champion'],
    values=winner['count'],
    name='Winner',
    textinfo='label+value'
)
trace2 = go.Pie(
    labels=runner['Runner-Up'],
    values=runner['count'],
    name='1st Runner',
    textinfo='label+value'
)
trace3 = go.Pie(
    labels=third['third_place'],
    values=third['count'],
    name='2nd Runner',
    textinfo='label+value'
)

fig = make_subplots(rows=1, cols=3, subplot_titles=['Winner', 'First Runnerup', 'Second Runnerup'], specs=[[{'type':'domain'}, {'type':'domain'}, {'type':'domain'}]])

fig.add_trace(trace1, row=1, col=1)
fig.add_trace(trace2, row=1, col=2)
fig.add_trace(trace3, row=1, col=3)
fig.update_layout(
    title_text=f"Champions and Runner-up Analysis",
    title_x=0.5,
    title_font=dict(size=24, family='Arial, sans-serif', color='darkblue'),
    showlegend=False
)
fig.show()
```



CONCLUSIONS

- Brazil won the most number of titles followed by Italy and Argentina.
- Argentina , Netherland and West Germany won Most number of Runner up titles.
- Germany was Second runnner up for most of the times.

Hosting Nation Playing in the Finals

```
host_in_finals = world_cup[(world_cup['Host'] == world_cup['Champion']) | (world_cup['Host'] == world_cup['Runner-Up'])]
host_winning_finals = world_cup[world_cup['Host'] == world_cup['Champion']]

print("Hosting Nation Playing in the Finals:")
print(tabulate(host_in_finals[['Year', 'Host', 'Champion', 'Runner-Up']].reset_index(drop=True), headers='keys', tablefmt='pretty'))

print()
print("Hosting Nation Winning the Finals:")
print(tabulate(host_winning_finals[['Year', 'Host', 'Champion']].reset_index(drop=True), headers='keys', tablefmt='pretty'))
```

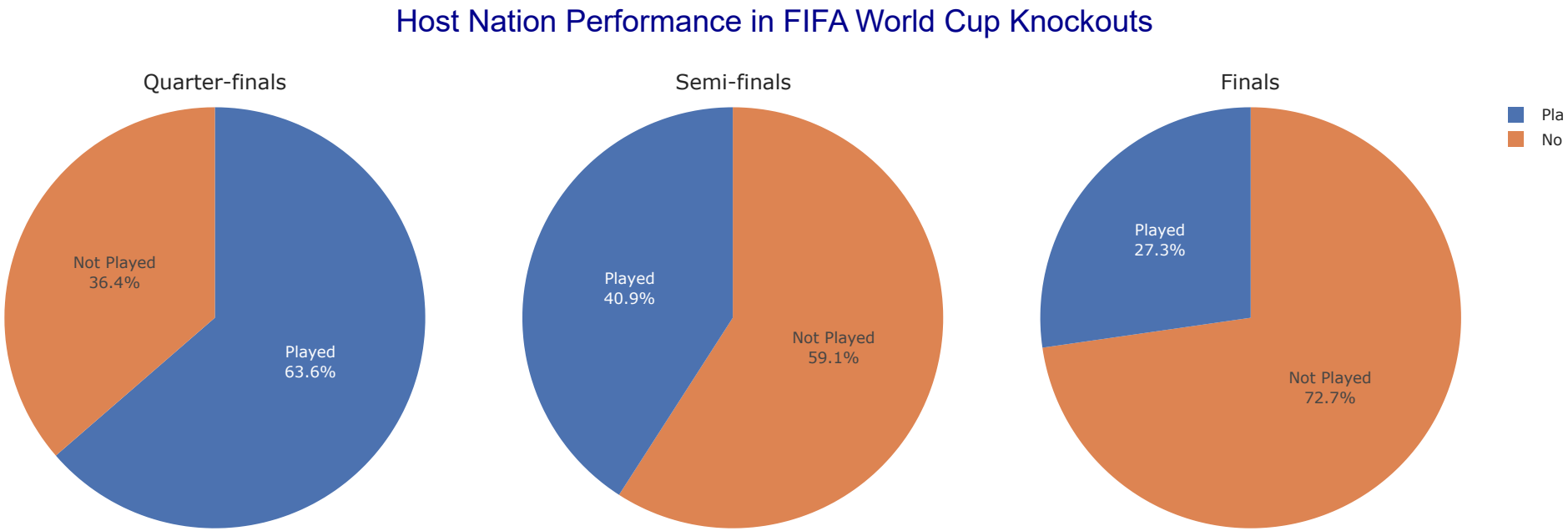
```
Hosting Nation Playing in the Finals:
+---+------+-----------+-----------+----------------+
|   | Year |   Host    | Champion  |   Runner-Up    |
+---+------+-----------+-----------+----------------+
| 0 | 1998 |  France   |  France   |     Brazil     |
| 1 | 1978 | Argentina | Argentina |  Netherlands   |
| 2 | 1966 |  England  |  England  |  West Germany  |
| 3 | 1958 |  Sweden   |  Brazil   |     Sweden     |
| 4 | 1950 |  Brazil   |  Uruguay  |     Brazil     |
| 5 | 1934 |   Italy   |   Italy   | Czechoslovakia |
| 6 | 1930 |  Uruguay  |  Uruguay  |   Argentina    |
+---+------+-----------+-----------+----------------+

Hosting Nation Winning the Finals:
+---+------+-----------+-----------+
|   | Year |   Host    | Champion  |
+---+------+-----------+-----------+
| 0 | 1998 |  France   |  France   |
| 1 | 1978 | Argentina | Argentina |
| 2 | 1966 |  England  |  England  |
| 3 | 1934 |   Italy   |   Italy   |
| 4 | 1930 |  Uruguay  |  Uruguay  |
+---+------+-----------+-----------+
```

Probability of Host Nation Playing the Knockouts

```
world_cup_hosts = world_cup[['Host','Year']]
knockout_matches = matches[matches['knockout']=='Knockout'][['home_team','away_team','Round','Year','winner']]

knockout_matches = knockout_matches.merge(world_cup_hosts, on='Year', how='left')
knockout_matches['host_played'] = (knockout_matches['home_team']==knockout_matches['Host']) | (knockout_matches['away_team']==knockout_matches['Host'])
```

```
total_world_cups = len(world_cup)
def counts(round):
    played = len(knockout_matches[(knockout_matches['Round']==round) & (knockout_matches['host_played'])])
    return played,total_world_cups-played

quarters_played, quarters_not_played = counts('Quarter-finals')
semis_played, semis_not_played = counts('Semi-finals')
finals_played, finals_not_played = counts('Final')
```

```
trace1 = go.Pie(
    labels=['Played', 'Not Played'],
    values=[quarters_played, quarters_not_played],
    textinfo='label+percent',
    name='Quarter-finals'
)
trace2 = go.Pie(
    labels=['Played', 'Not Played'],
    values=[semis_played, semis_not_played],
    textinfo='label+percent',
    name='Semi-finals'
)
trace3 = go.Pie(
    labels=['Played', 'Not Played'],
    values=[finals_played, finals_not_played],
    textinfo='label+percent',
    name='Finals'
)

fig = make_subplots(rows=1, cols=3,
    subplot_titles=['Quarter-finals','Semi-finals','Finals'],
    specs=[[{'type':'domain'}, {'type':'domain'}, {'type':'domain'}]])

fig.add_trace(trace1, row=1, col=1)
fig.add_trace(trace2, row=1, col=2)
fig.add_trace(trace3, row=1, col=3)
fig.update_layout(
    title_text='Host Nation Performance in FIFA World Cup Knockouts',
    title_x=0.5,
    title_font=dict(size=24, family='Arial, sans-serif', color='darkblue'),
)

fig.show()
```

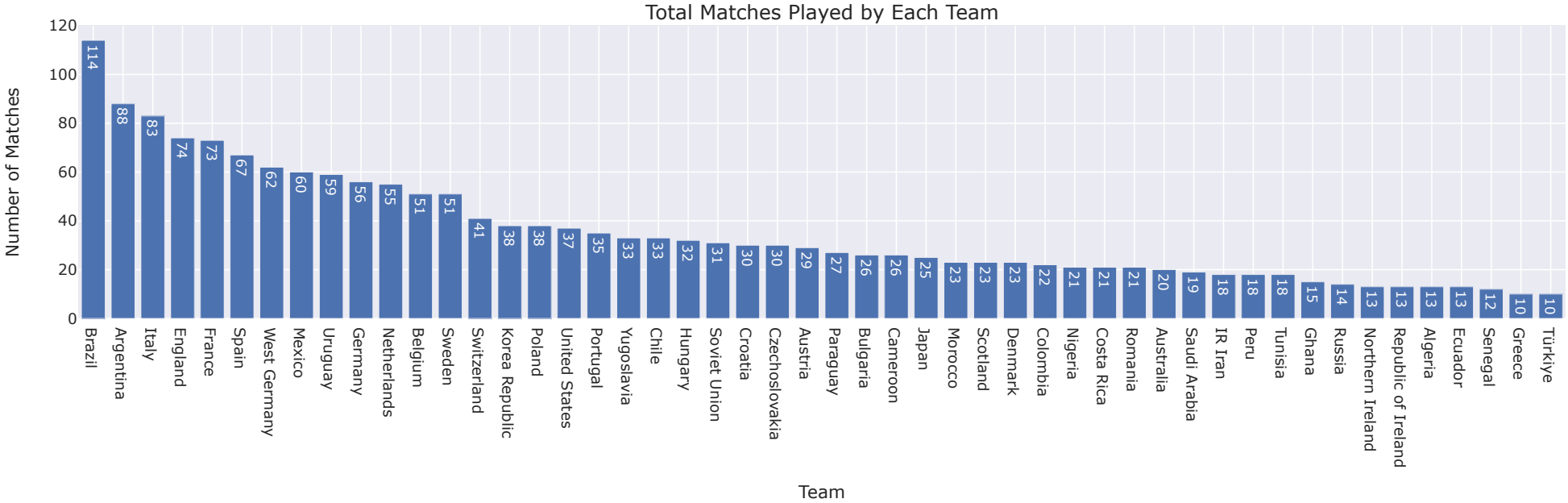


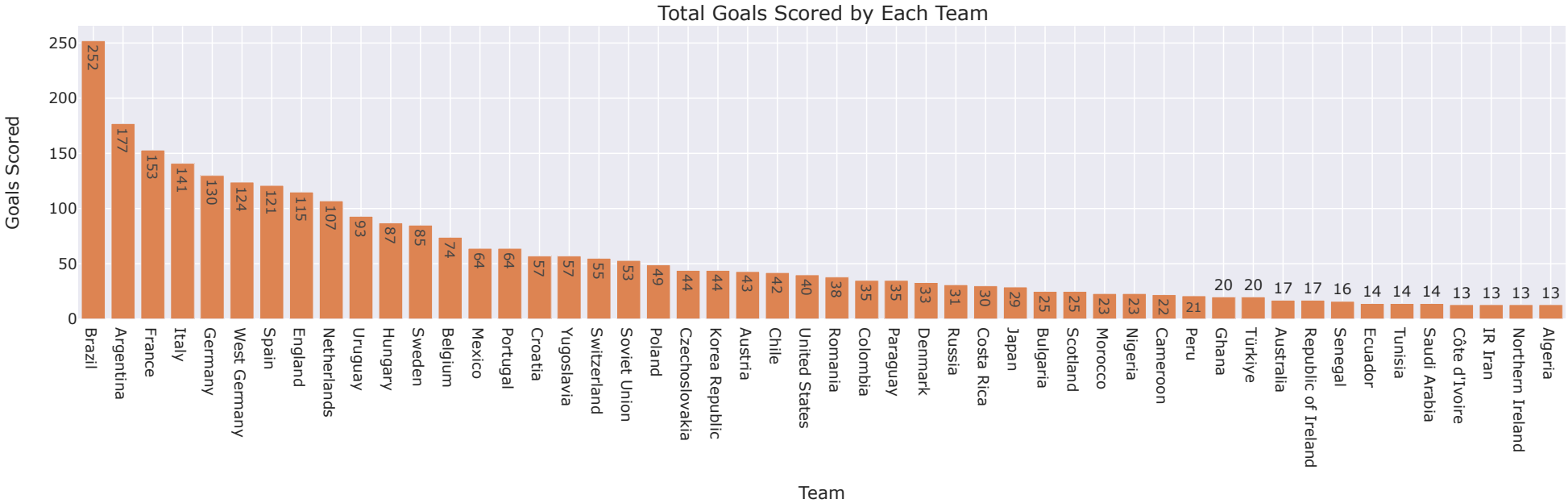
Team Wise Matches and Goals Analysis

```
home_team_matches = matches['home_team'].value_counts()
away_team_matches = matches['away_team'].value_counts()
total_team_matches = home_team_matches.add(away_team_matches, fill_value=0).sort_values(ascending=False)
d1_50 = total_team_matches.head(50)

home_goals = matches.groupby('home_team')['home_total'].sum().sort_values(ascending=False)
away_goals = matches.groupby('away_team')['away_total'].sum().sort_values(ascending=False)
total_goals = home_goals.add(away_goals, fill_value=0).sort_values(ascending=False)
d2_50 = total_goals.head(50)

avg_goals = total_goals.divide(total_team_matches, fill_value=0).sort_values(ascending=False)
d3_50 = avg_goals.head(50)

trace1 = go.Bar(x=d1_50.index, y=d1_50.values,
            text=d1_50.values,
            name='Matches Played'
```

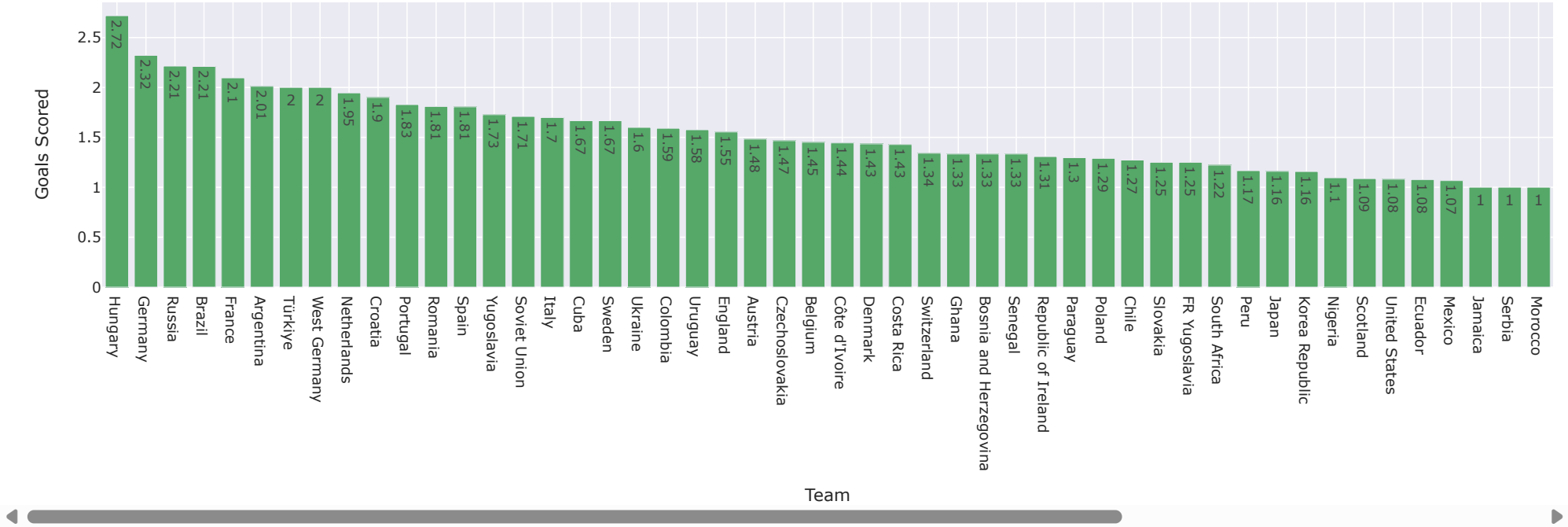
```
)
trace2 = go.Bar(x=d2_50.index, y=d2_50.values,
            text=d2_50.values,
            name='Goals Scored'
)
trace3 = go.Bar(x=d3_50.index, y=d3_50.values,
            text=np.round(d3_50.values,2),
            name='Goals Per Match'
)

fig = make_subplots(rows=3, cols=1, subplot_titles=(
    'Total Matches Played by Each Team',
    'Total Goals Scored by Each Team',
    'Average Goals per Match for Each Team'
))

fig.add_trace(trace1, row=1, col=1)
fig.add_trace(trace2, row=2, col=1)
fig.add_trace(trace3, row=3, col=1)

fig.update_layout(
    title_text='Team Wise Matches and Goals Analysis',
    title_x=0.5,
    title_font=dict(size=24, family='Arial, sans-serif', color='darkblue'),
    xaxis_title='Team',
    yaxis_title='Number of Matches',
    xaxis2_title='Team',
    yaxis2_title='Goals Scored',
    xaxis3_title='Team',
    yaxis3_title='Goals Scored',
    showlegend=False,
    height=1350)
fig.show()
```

Team Wise Matches and Goals Analysis





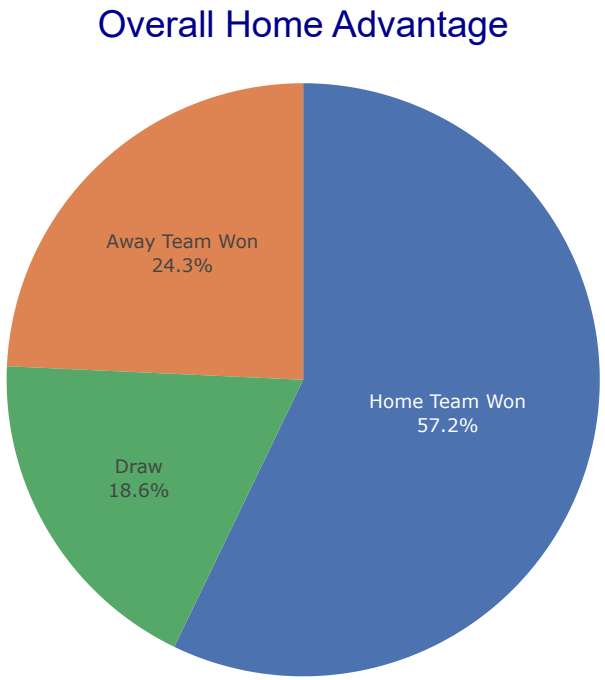
Average Goals per Match for Each Team



Overall Home Advantage

```
matches['result'] = matches.apply(lambda row: 'Home Team Won' if row['home_total'] > row['away_total'] else ('Away Team Won' if row['home_total'] < row['away_total'] else '
```

```
wins = matches['result'].value_counts().reset_index()

fig = px.pie(data_frame=wins,names='result', values='count')
fig.update_traces(textinfo='label+percent')
fig.update_layout(title_text='Overall Home Advantage',
                  title_x=0.5,
                  title_font=dict(size=24, family='Arial, sans-serif', color='darkblue'),
                  showlegend=False)
fig.show()
```

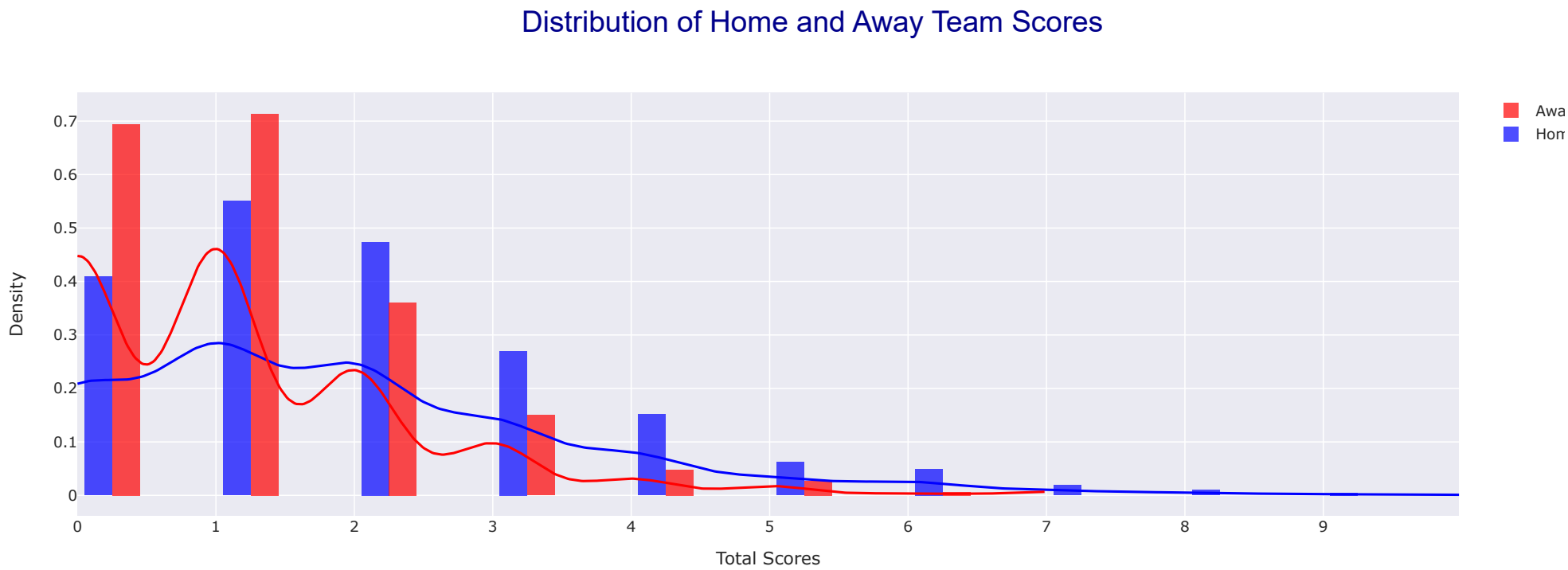


Conclusion:

- Percentage of Visiting team winning the match is very low (only 24.3%)
- Indicating a Clear Advantage for the Home Team

Distribution of Home and Away Team Scores

```
fig = create_distplot(
    [matches['home_total'], matches['away_total']],
    group_labels=['Home Team', 'Away Team'],
    bin_size=0.5,
    show_hist=True,
    show_rug=False,
    colors=['blue', 'red']
)

fig.update_layout(
    xaxis_title='Total Scores',
    yaxis_title='Density',
    barmode='group',
    title='Distribution of Home and Away Team Scores',
    title_x=0.5,
    title_font=dict(size=24, family='Arial, sans-serif', color='darkblue')
)
fig.show()
```



Conclusions:

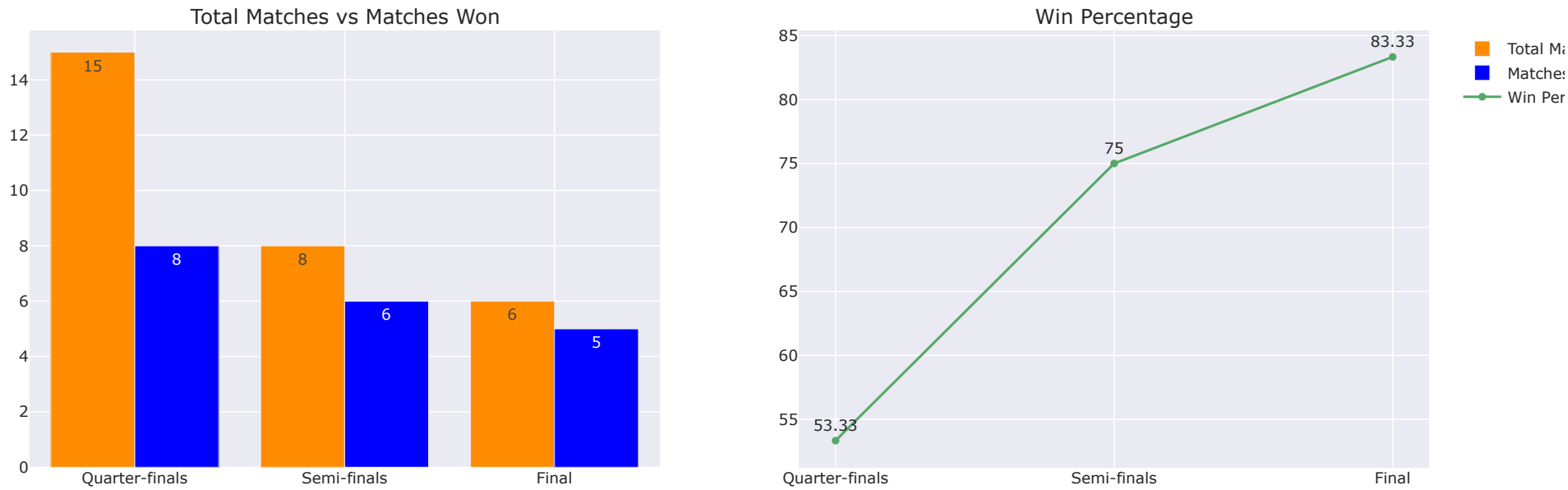
- Home teams have a slightly broader distribution of scores compared to away teams. This means home teams are more likely to have a higher range of scores.
- Away teams tend to score fewer goals on average, with most of their scores clustered around 0, 1, and 2.
- Away team density curve is more skewed towards the lower end of the score range, indicating that away teams generally score fewer goals and therefore they have a lower chance of winning.

Team Performance in Knockouts

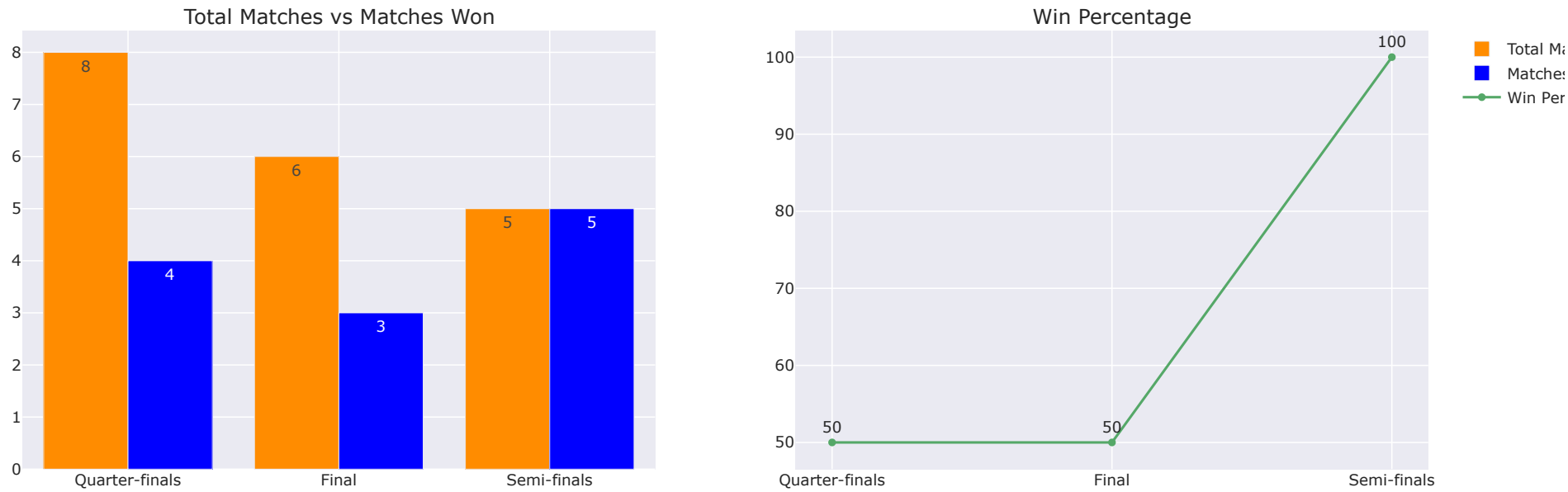
```
def calculate_knockout_performance(team):
    data = matches[((matches['home_team']==team) | (matches['away_team']==team)) & (matches['knockout']=='Knockout')]

    df1 = data['Round'].value_counts().reset_index()
    df1.columns = ['Round','total_matches']

    # there will be no drawn matches in knockouts
    df2 = data[data['winner']==team]['Round'].value_counts().reset_index()
    df2.columns = ['Round','matches_won']

    df = pd.merge(df1,df2,on='Round')
    df['win_percent'] = df['matches_won']/df['total_matches']*100
    return df
```

```
def knockout_performance(team):
    performance_df  = calculate_knockout_performance(team)

    bar_total_matches = go.Bar(
        x=performance_df['Round'],
        y=performance_df['total_matches'],
        name='Total Matches',
        text=performance_df['total_matches'],
        marker=dict(color='darkorange'))

    bar_matches_won = go.Bar(
        x=performance_df['Round'],
        y=performance_df['matches_won'],
        name='Matches Won',
        text=performance_df['matches_won'],
        marker=dict(color='blue'))

    line_win_percent = go.Scatter(
        x=performance_df['Round'],
        y=performance_df['win_percent'],
        name='Win Percentage',
        mode='lines+markers+text',
        text=np.round(performance_df['win_percent'],2),
        textposition='top center',
        hovertemplate='Round: %{x}<br>Win Percentage: %{y:.2f}%'
    )

    fig = make_subplots(
        rows=1, cols=2,
        subplot_titles=['Total Matches vs Matches Won', 'Win Percentage'],
    )
    fig.add_trace(bar_total_matches, row=1, col=1)
    fig.add_trace(bar_matches_won, row=1, col=1)
    fig.add_trace(line_win_percent, row=1, col=2)

    fig.update_layout(
        title_text=f"Performance of {team} in Knockout Rounds",
        title_x=0.5,
        title_font=dict(size=24, family='Arial, sans-serif', color='darkblue'),
    )
    fig.show()
```

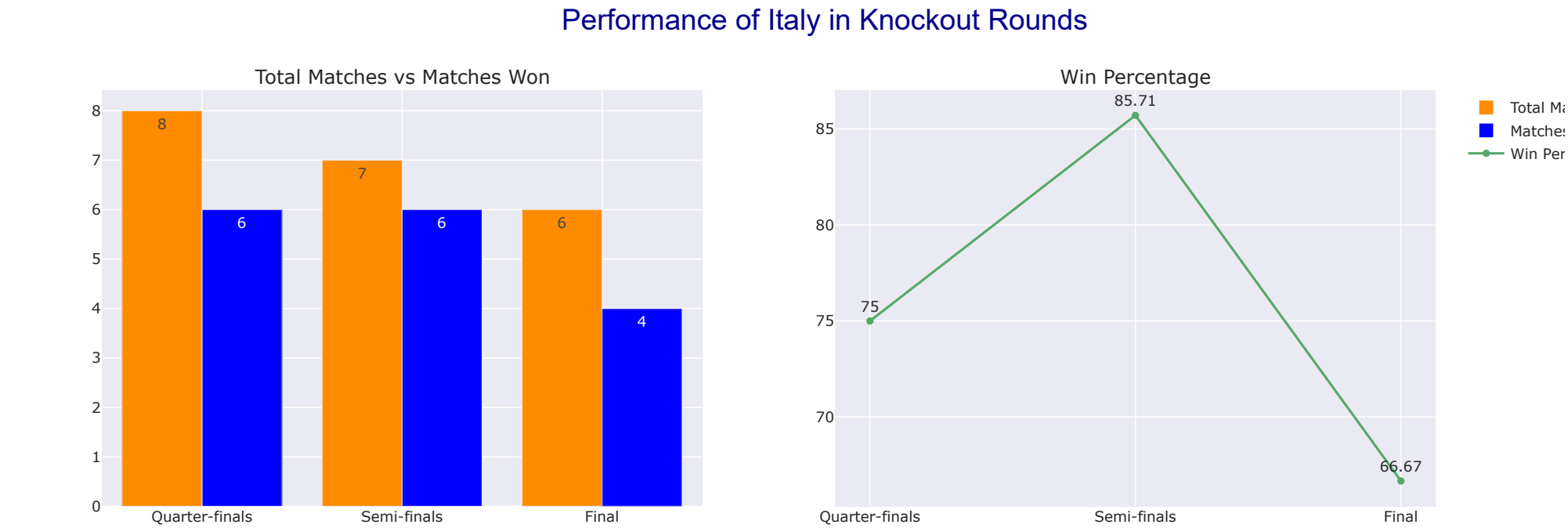
```
knockout_performance('Brazil')
```



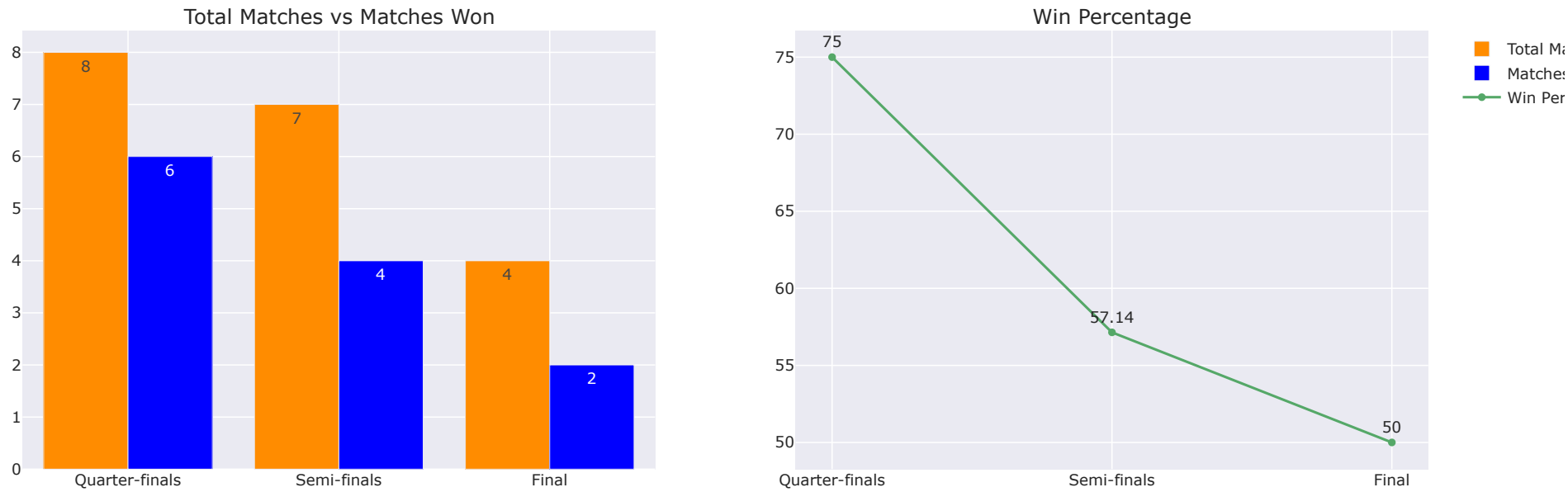

```
knockout_performance('Argentina')
```




```
knockout_performance('Italy')
```




```
knockout_performance('France')
```



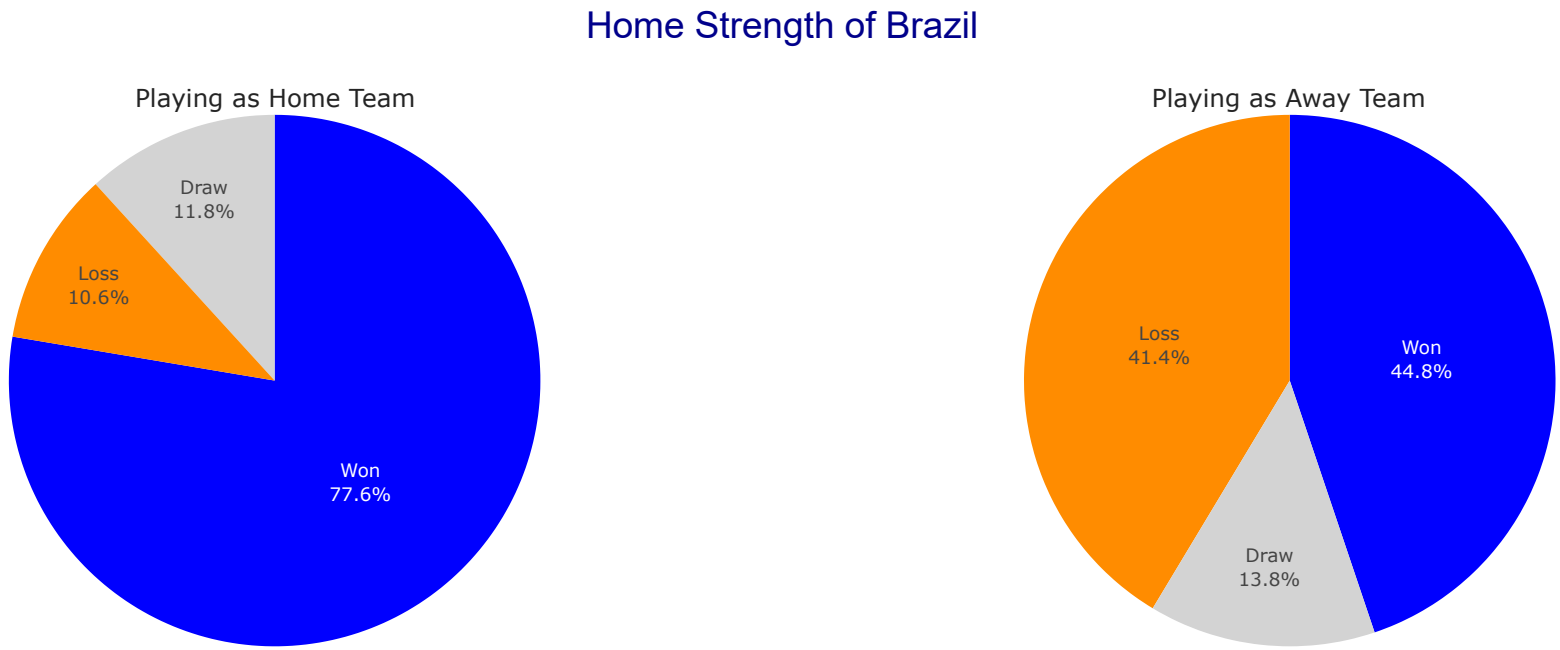
CONCLUSIONS

- Brazil shows consistent improvement as they advance through the knockouts, indicating a strong finish.
- Teams facing Italy in knockouts can expect a challenging match.
- If Argentina advances to the Semi-finals, they win it for sure. But struggles in both the Quarter-finals and Finals with a 50% win rate.
- France has a strong start (75% in Quarter-finals) but their performance decreases in the later stages of the knockouts.

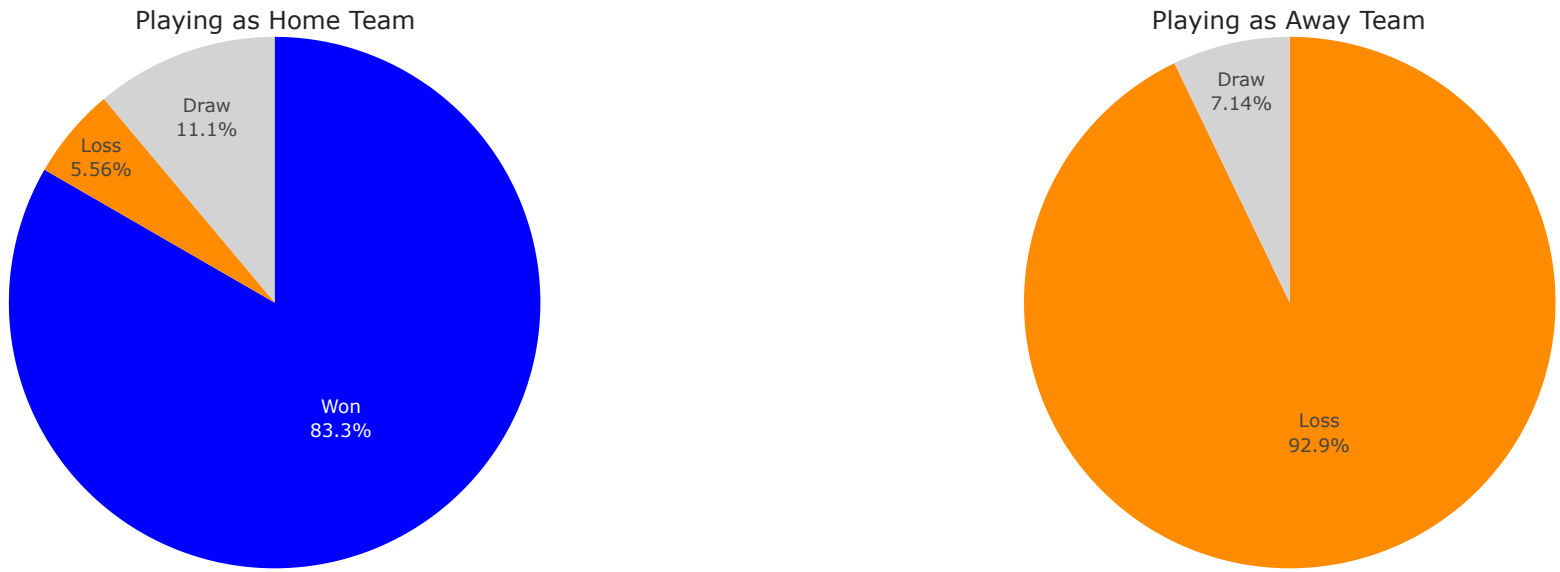
Home Strength for Each Team

```
def home_strength(team):
    home = matches[matches['home_team']==team]['result'].value_counts().reset_index()
    away = matches[matches['away_team']==team]['result'].value_counts().reset_index()

    home['result'] = home['result'].apply(lambda x: 'Won' if x=='Home Team Won' else ('Loss' if x=='Away Team Won' else x))
    away['result'] = away['result'].apply(lambda x: 'Won' if x=='Away Team Won' else ('Loss' if x=='Home Team Won' else x))

    home_colors = ['blue' if label == 'Won' else 'darkorange' if label == 'Loss' else 'lightgrey' for label in home['result']]
    away_colors = ['blue' if label == 'Won' else 'darkorange' if label == 'Loss' else 'lightgrey' for label in away['result']]

    trace1 = go.Pie(
        labels=home['result'],
        values=home['count'],
        name='As Home Team',
        textinfo='label+percent',
        marker=dict(colors=home_colors)
    )
    trace2 = go.Pie(
        labels=away['result'],
        values=away['count'],
        name='As Away Team',
        textinfo='label+percent',
        marker=dict(colors=away_colors)
    )

    fig = make_subplots(rows=1, cols=2, subplot_titles=['Playing as Home Team', 'Playing as Away Team'], specs=[[{'type':'domain'}, {'type':'domain'}]])

    fig.add_trace(trace1, row=1, col=1)
    fig.add_trace(trace2, row=1, col=2)
    fig.update_layout(
        title_text=f"Home Strength of {team}",
        title_x=0.5,
        title_font=dict(size=24, family='Arial, sans-serif', color='darkblue'),
        showlegend=False
    )
    fig.show()
```

```
home_strength('Brazil')
```




```
home_strength('Hungary')
```



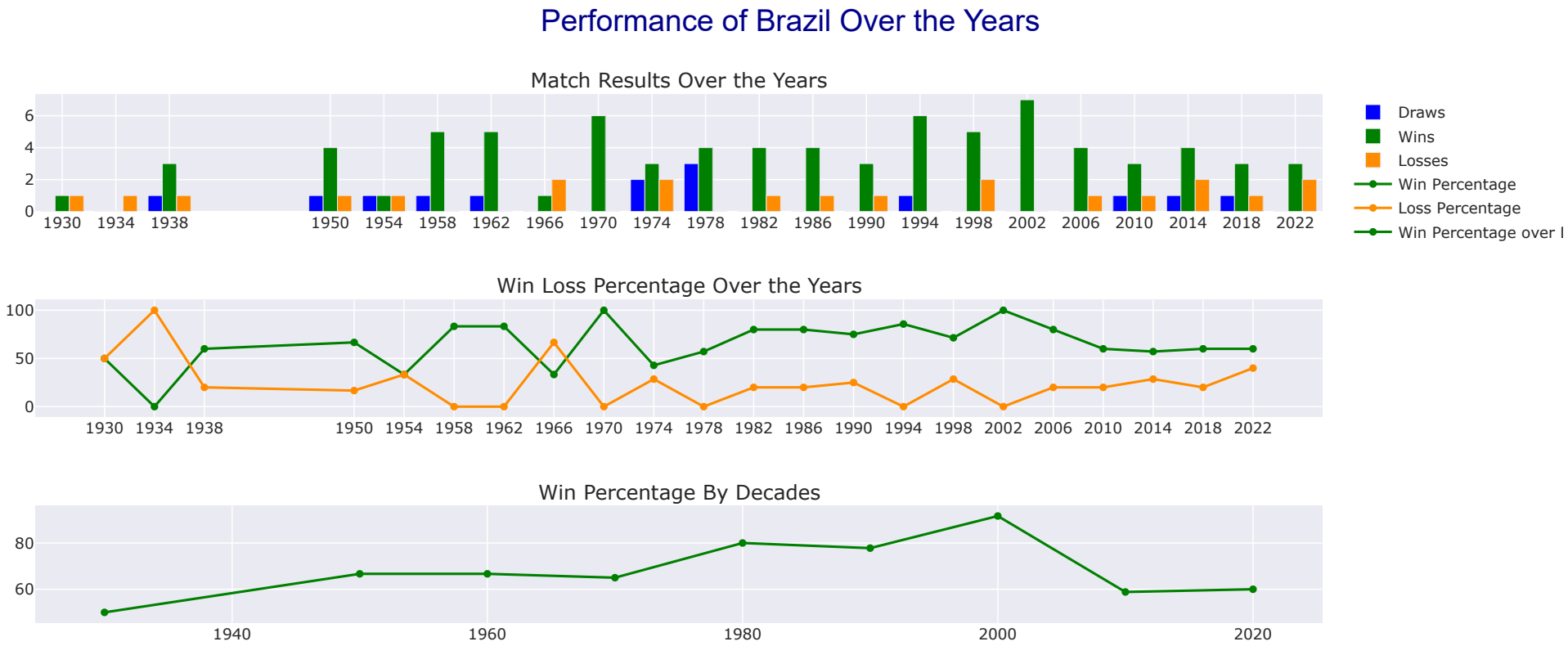
Conclusion:

- Brazil has demonstrated a strong performance at home, losing only 10.6% of their matches when playing as the home team. This indicates their dominance and competitive advantage when hosting matches on their home turf.
- Hungary has shown exceptional strength in home matches, losing only 5.56% of their games. This highlights their competitive edge when playing at home. However, Hungary has not secured a single victory in away matches, indicating a significant drop in performance when playing outside their home country. This emphasizes the importance of home advantage for Hungary's team.

Team Performance over the Years

```
def team_performance_trend(team):
    team_matches = matches[(matches['home_team'] == team) | (matches['away_team'] == team)]
    team_matches['result'] = team_matches['winner'].apply(lambda x: 'Win' if x==team else (x if x=='Draw' else 'Loss'))

    yearly_performance = team_matches.groupby(['Year', 'result']).size().unstack().fillna(0)
    yearly_performance['Total Matches'] = yearly_performance.sum(axis=1)
    yearly_performance['Win Percentage'] = yearly_performance['Win'] / yearly_performance['Total Matches'] * 100
    yearly_performance['Loss Percentage'] = yearly_performance['Loss'] / yearly_performance['Total Matches'] * 100

    team_matches['decade'] = (team_matches['Year'] // 10) * 10
    team_matches['win'] = team_matches.apply(lambda row: 1 if row['winner']==team else 0, axis=1)
    win_percentage_od = team_matches.groupby('decade')['win'].mean() * 100
    win_percentage_od_trace = go.Scatter(x=win_percentage_od.index, y=win_percentage_od.values, name='Win Percentage over Decades', mode='lines+markers',line=dict(color='gr


    draws = go.Bar(x=yearly_performance.index, y=yearly_performance['Draw'], name='Draws', marker=dict(color='blue'))
    wins = go.Bar(x=yearly_performance.index, y=yearly_performance['Win'], name='Wins', marker=dict(color='green'))
    losses = go.Bar(x=yearly_performance.index, y=yearly_performance['Loss'], name='Losses', marker=dict(color='darkorange'))

    win_percentage = go.Scatter(x=yearly_performance.index, y=yearly_performance['Win Percentage'], name='Win Percentage', mode='lines+markers', line=dict(color='green'))
    loss_percentage = go.Scatter(x=yearly_performance.index, y=yearly_performance['Loss Percentage'], name='Loss Percentage', mode='lines+markers',line=dict(color='darkoran

    fig = make_subplots(rows=3, cols=1, subplot_titles=['Match Results Over the Years', 'Win Loss Percentage Over the Years','Win Percentage By Decades'])

    fig.add_trace(draws, row=1, col=1)
    fig.add_trace(wins, row=1, col=1)
    fig.add_trace(losses, row=1, col=1)
    fig.add_trace(win_percentage, row=2, col=1)
    fig.add_trace(loss_percentage, row=2, col=1)
    fig.add_trace(win_percentage_od_trace, row=3, col=1)

    fig.update_layout(
        title_text=f"Performance of {team} Over the Years",
        title_x=0.5,
        title_font=dict(size=24, family='Arial, sans-serif', color='darkblue'),
        height=600,
        xaxis=dict(tickvals=team_matches['Year'].unique()),
        xaxis2=dict(tickvals=world_cup['Year'].unique())
    )
    fig.show()
```

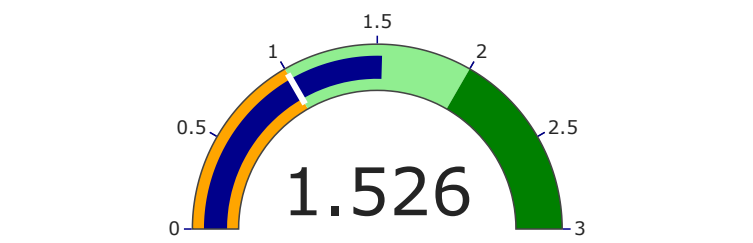
```
team_performance_trend('Brazil')
```



Goals Scored Vs Conceded

```
def goals_conceded_vs_scored(team):
    team_matches = matches[(matches['home_team'] == team) | (matches['away_team'] == team)]

    team_matches['goals_scored'] = team_matches.apply(lambda row: row['home_total'] if row['home_team'] == team else row['away_total'], axis=1)
    team_matches['goals_conceded'] = team_matches.apply(lambda row: row['away_total'] if row['home_team'] == team else row['home_total'], axis=1)

    total_goals_scored = team_matches['goals_scored'].sum()
    total_goals_conceded = team_matches['goals_conceded'].sum()
    goals_ratio = total_goals_scored / total_goals_conceded if total_goals_conceded != 0 else float('inf')

    fig = go.Figure()
    fig.add_trace(go.Indicator(
        mode = "gauge+number",
        value = goals_ratio,
        gauge = {'axis': {'range': [None, 3], 'tickwidth': 1, 'tickcolor': "darkblue"},
                 'bar': {'color': "darkblue"},
                 'steps': [
                     {'range': [0, 1], 'color': 'orange'},
                     {'range': [1, 2], 'color': 'lightgreen'},
                     {'range': [2, 3], 'color': 'green'}],
                 'threshold': {'line': {'color': "white", 'width': 4}, 'thickness': 0.75, 'value': 1}}))

    fig.update_layout(
        title_text=f"Goals Scored to Conceded Ratio for {team}",
        title_x=0.5,
        title_font=dict(size=24, family='Arial, sans-serif', color='darkblue'),
        height=300, width=600
    )
    fig.show()
```

```
goals_conceded_vs_scored('Hungary')
```



Adding Goal Timings

```
matches['home_goal'][0]
```

```
'Ángel Di María - 36|Lionel Messi - 108'
```

```
pattern = r'\b(\d+)\b'

def home_goal_time(row):
    if pd.notna(row['home_goal']):
        times = re.findall(pattern,str(row['home_goal']))
        return ', '.join(times)
    return ''
def away_goal_time(row):
    if pd.notna(row['away_goal']):
        times = re.findall(pattern,str(row['away_goal']))
        return ', '.join(times)
    return ''

matches['home_goal_time'] = matches.apply(home_goal_time, axis=1)
matches['away_goal_time'] = matches.apply(away_goal_time, axis=1)
```

```
matches.iloc[:5,-5:]
```

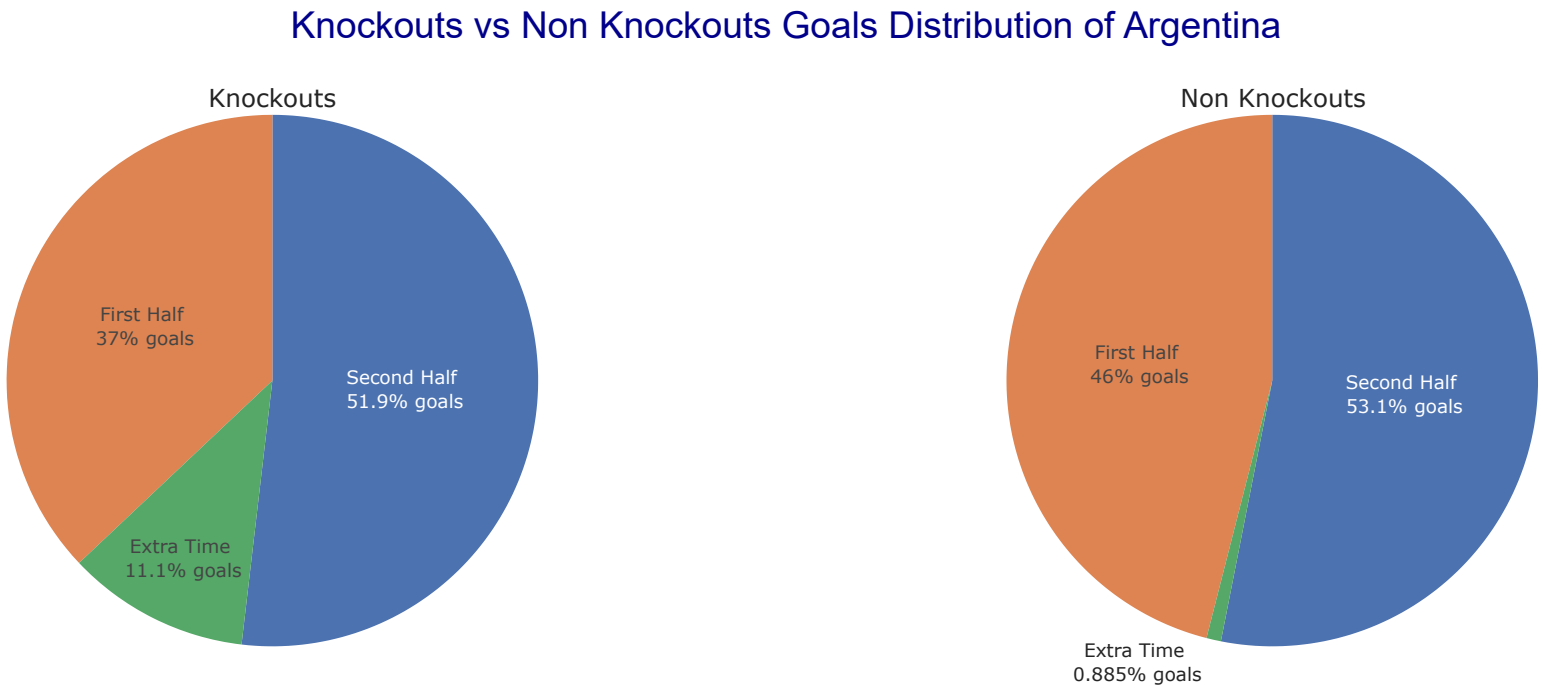
	winner	knockout	result	home_goal_time	away_goal_time
0	Argentina	Knockout	Home Team Won	36, 108	81
1	Croatia	Non-Knockout	Home Team Won	7, 42	9
2	France	Knockout	Home Team Won	5, 79	
3	Argentina	Knockout	Home Team Won	39, 69	
4	Morocco	Knockout	Home Team Won	42	

Strategy of Team in Knockout VS Non-Knockout matches

```
def strategy(team):
    def time(row):
        if row['time']<=45:
            return 'First Half'
        elif row['time']<=90:
            return 'Second Half'
        else:
            return 'Extra Time'

    # Separating Knockout and Non-Knockouts
    home_k = matches[(matches['home_team']==team)&(matches['knockout']=='Knockout')]
    away_k = matches[(matches['away_team']==team)&(matches['knockout']=='Knockout')]

    home_nk = matches[(matches['home_team']==team)&(matches['knockout']=='Non-Knockout')]
    away_nk = matches[(matches['away_team']==team)&(matches['knockout']=='Non-Knockout')]

    # Knockout Matches
    hkgt = home_k['home_goal_time'].str.split(',').explode().reset_index()
    akgt = away_k['away_goal_time'].str.split(',').explode().reset_index()

    knockouts_gt = pd.concat([hkgt,akgt], axis=0, ignore_index=True)
    knockouts_gt.replace(r'^\s*$', 0, regex=True, inplace=True) # empty values
    knockouts_gt.fillna(0, inplace=True)

    knockouts_gt['home_goal_time'] = knockouts_gt['home_goal_time'].astype(int)
    knockouts_gt['away_goal_time'] = knockouts_gt['away_goal_time'].astype(int)

    knockouts_gt['time'] = knockouts_gt['home_goal_time'] + knockouts_gt['away_goal_time']
    knockouts_gt = knockouts_gt[knockouts_gt['time']!=0]

    knockouts_gt['half'] = knockouts_gt.apply(time, axis=1)

    # Non Knockout Matches
    hnkgt = home_nk['home_goal_time'].str.split(',').explode().reset_index()
    ankgt = away_nk['away_goal_time'].str.split(',').explode().reset_index()

    non_knockouts_gt = pd.concat([hnkgt,ankgt], axis=0, ignore_index=True)
    non_knockouts_gt.replace(r'^\s*$', 0, regex=True, inplace=True)
    non_knockouts_gt.fillna(0, inplace=True)
    non_knockouts_gt.drop(columns=['index'], inplace=True)

    non_knockouts_gt['home_goal_time'] = non_knockouts_gt['home_goal_time'].astype(int)
    non_knockouts_gt['away_goal_time'] = non_knockouts_gt['away_goal_time'].astype(int)

    non_knockouts_gt['time'] = non_knockouts_gt['home_goal_time'] + non_knockouts_gt['away_goal_time']
    non_knockouts_gt = non_knockouts_gt[non_knockouts_gt['time']!=0]

    non_knockouts_gt['half'] = non_knockouts_gt.apply(time, axis=1)
```

```
    trace1 = go.Pie(labels=knockouts_gt['half'].value_counts().index,
                    values=knockouts_gt['half'].value_counts(),
                    name='Knockouts',
                    texttemplate='%{label} <br>%{percent} goals')

    trace2 = go.Pie(labels=non_knockouts_gt['half'].value_counts().index,
                    values=non_knockouts_gt['half'].value_counts(),
                    name='Non Knockouts',
                    texttemplate='%{label} <br>%{percent} goals')

    fig = make_subplots(rows=1, cols=2, subplot_titles=['Knockouts', 'Non Knockouts'], specs=[[{'type':'domain'}, {'type':'domain'}]])

    fig.add_trace(trace1, row=1, col=1)
    fig.add_trace(trace2, row=1, col=2)
    fig.update_layout(title_text=f"Knockouts vs Non Knockouts Goals Distribution of {team}",
                      title_x=0.5,
                      title_font=dict(size=24, family='Arial, sans-serif', color='darkblue'),
                      showlegend=False)
    fig.show()
```

```
strategy('Argentina')
```



Cleaning Substitute Data

```
matches['home_substitute_in_long'][0]
```

```
'['64&rsquor;|2:0|Marcos Acuña|for Ángel Di María', '91&rsquor;|2:2|Gonzalo Montiel|for Nahuel Molina', '102&rsquor;|2:2|Leandro Paredes|for Rodrigo De Paul', '103&rsquor;|2:2|Lautaro Martínez|for Julián Álvarez', '116&rsquor;|3:2|Germán Pezzella|for Alexis Mac Allister', '120+1&rsquor;|3:3|Paulo Dybala|for Nicolás Tagliafico']'
```

```
def substitute_data(row):
    if pd.isna(row):
        return []
    pattern = r"(\d{1,3}(?:\+\d{1,2})?)&rsquor;\|\d:\d\|([^|]+)\|for ([^']+)"
    x = re.findall(pattern, row)
    result = []
    for match in x:
        try:
            minute = int(match[0].split('+')[0]) + int(match[0].split('+')[1]) if '+' in match[0] else int(match[0])
        except ValueError:
            minute = match[0]
        result.append({'minute': minute, 'player_in': match[1], 'player_out': match[2]})
    return result

matches['home_substitutions'] = matches['home_substitute_in_long'].apply(substitute_data)
matches['away_substitutions'] = matches['away_substitute_in_long'].apply(substitute_data)
```

```
matches['home_substitutions'][0]
```

```
[{'minute': 64, 'player_in': 'Marcos Acuña', 'player_out': 'Ángel Di María'},
 {'minute': 91, 'player_in': 'Gonzalo Montiel', 'player_out': 'Nahuel Molina'},
 {'minute': 102,
  'player_in': 'Leandro Paredes',
  'player_out': 'Rodrigo De Paul'},
 {'minute': 103,
  'player_in': 'Lautaro Martínez',
  'player_out': 'Julián Álvarez'},
 {'minute': 116,
  'player_in': 'Germán Pezzella',
  'player_out': 'Alexis Mac Allister'},
 {'minute': 121,
  'player_in': 'Paulo Dybala',
  'player_out': 'Nicolás Tagliafico'}]
```

Cleaning Goal Data

```
matches['home_goal'][0].split('|')
```

```
['Ángel Di María · 36', 'Lionel Messi · 108']
```

```
def goal_data(row):
    if pd.isna(row):
        return []
    goals = row.split('|')
    result = []
    for goal in goals:
        x = goal.split('·') # alt+0183 => ·
        if len(x)==2:
            try:
                result.append({'minute':int(x[1].strip()), 'scorer':x[0].strip()})
            except:
                pass
    return result

matches['home_goal_details'] = matches['home_goal'].apply(goal_data)
matches['away_goal_details'] = matches['away_goal'].apply(goal_data)
```

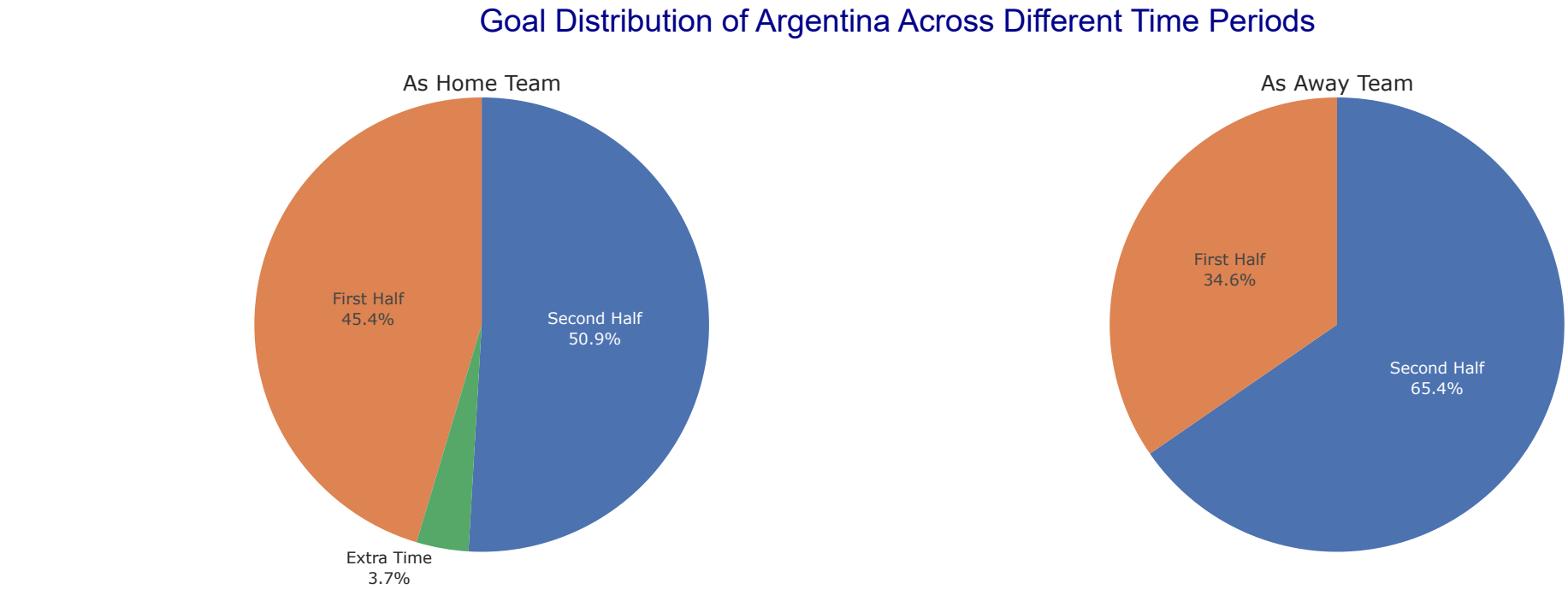
```
matches['home_goal_details'][0]
```

```
[{'minute': 36, 'scorer': 'Ángel Di María'},
 {'minute': 108, 'scorer': 'Lionel Messi'}]
```

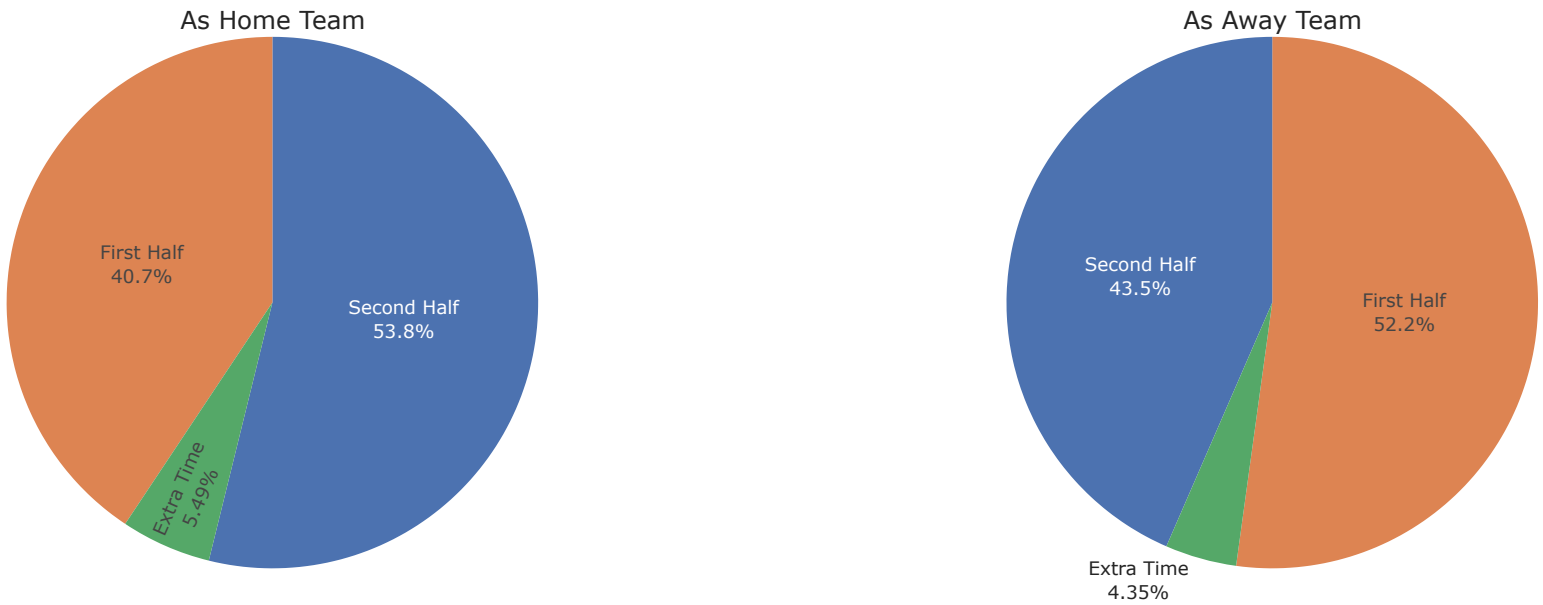
Team Goal Distribution Home Vs Away

```
def goal_distribution(team):
    def time(minute):
        if minute <= 45:
            return 'First Half'
        elif minute <= 90:
            return 'Second Half'
        else:
            return 'Extra Time'
    team_matches = matches[(matches['home_team']==team) | (matches['away_team']==team)]

    home_goals = []
    away_goals = []
    for _, row in team_matches.iterrows():
        if row['home_team'] == team:
            if row['home_goal_details']:
                for goal in row['home_goal_details']:
                    home_goals.append({'minute': goal['minute'], 'team': row['home_team'], 'opponent': row['away_team'], 'match_date': row['Date']})
        if row['away_team'] == team:
            if row['away_goal_details']:
                for goal in row['away_goal_details']:
                    away_goals.append({'minute': goal['minute'], 'team': row['away_team'], 'opponent': row['home_team'], 'match_date': row['Date']})

    home_goals_df = pd.DataFrame(home_goals)
    away_goals_df = pd.DataFrame(away_goals)

    home_goals_df['half'] = home_goals_df['minute'].apply(time)
    away_goals_df['half'] = away_goals_df['minute'].apply(time)

    fig = make_subplots(rows=1, cols=2, subplot_titles=['As Home Team', 'As Away Team'], specs=[[{'type': 'domain'}, {'type': 'domain'}]])
    trace1 = go.Pie(labels=home_goals_df['half'].value_counts().index,
                    values=home_goals_df['half'].value_counts(),
                    name='As Home Team',
                    textinfo='label+percent'
                    )
    trace2 = go.Pie(labels=away_goals_df['half'].value_counts().index,
                    values=away_goals_df['half'].value_counts(),
                    name='As Home Team',
                    textinfo='label+percent'
                    )
    fig.add_trace(trace1, row=1, col=1)
    fig.add_trace(trace2, row=1, col=2)
    fig.update_layout(
        title_text=f"Goal Distribution of {team} Across Different Time Periods",
        title_x=0.5,
        title_font=dict(size=24, family='Arial, sans-serif', color='darkblue'),
        showlegend=False
    )
    fig.show()
```

```
goal_distribution('Argentina')
```




```
goal_distribution('Italy')
```

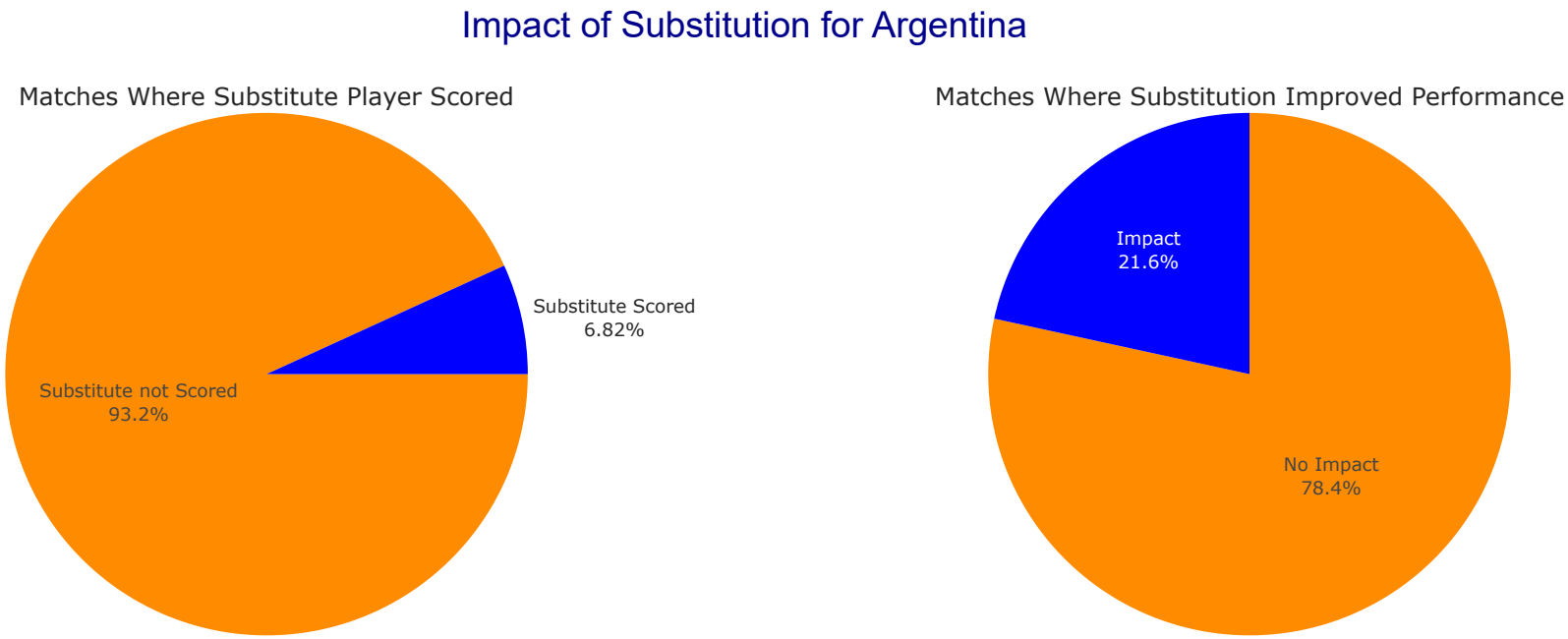


Substitution Impact

```
def substitute_summary(team):
    team_matches = matches[(matches['home_team']==team) | (matches['away_team']==team)]

    # goals by substituted player
    def susbstitute_goals(subs,goals):
        if not subs:
            return False
        for sub in subs:
            if any(goal['scorer']==sub['player_in'] for goal in goals):
                return True
        return False
    def check_substitute_goals(row):
        if row['home_team']==team:
            return susbstitute_goals(row['home_substitutions'],row['home_goal_details'])
        elif row['away_team']==team:
            return susbstitute_goals(row['away_substitutions'],row['away_goal_details'])
        return False

    # substitution impact on goal scoring
    def check_substitution_impact(row):
        if row['home_team']==team:
            subs = row['home_substitutions']
            goals_before = [goal for goal in row['home_goal_details'] if goal['minute'] <= subs[0]['minute']] if subs else []
            goals_after = [goal for goal in row['home_goal_details'] if goal['minute'] > subs[0]['minute']] if subs else []
        elif row['away_team']==team:
            subs = row['away_substitutions']
            goals_before = [goal for goal in row['away_goal_details'] if goal['minute'] <= subs[0]['minute']] if subs else []
            goals_after = [goal for goal in row['away_goal_details'] if goal['minute'] > subs[0]['minute']] if subs else []
        else:
            return False
        return len(goals_after)>len(goals_before)

    # For each match checking the impact of substitute
    team_matches['substitute_goals'] = team_matches.apply(lambda row: check_substitute_goals(row),axis=1)
    team_matches['substitution_impact'] = team_matches.apply(lambda row: check_substitution_impact(row),axis=1)

    # Match wise summary of Substitution Impact
    match_wise_summary = []
    for index,row in team_matches.iterrows():
        match_summary = {
            'Match' : f"{row['home_team']} vs {row['away_team']}",
            'Date' : row['Date'],
            'Substitute goals' : row['substitute_goals'],
            'Substitution impact' : row['substitution_impact']
        }
        match_wise_summary.append(match_summary)
    match_wise_summary = pd.DataFrame(match_wise_summary)

    match_wise_summary['Substitution impact'] = match_wise_summary['Substitution impact'].map({True: 'Impact', False: 'No Impact'})
```

```
    match_wise_summary['Substitute goals'] = match_wise_summary['Substitute goals'].map({True: 'Substitute Scored', False: 'Substitute not Scored'})

    fig = make_subplots(rows=1, cols=2, subplot_titles=['Matches Where Substitute Player Scored', 'Matches Where Substitution Improved Performance'], specs=[[{'type': 'doma
    trace1 = go.Pie(labels=match_wise_summary['Substitute goals'].value_counts().index,
                    values=match_wise_summary['Substitute goals'].value_counts(),
                    name='Substitute Goals',
                    textinfo='label+percent',
                    marker=dict(colors=['darkorange','blue']),
                    rotation=90)
    trace2 = go.Pie(labels=match_wise_summary['Substitution impact'].value_counts().index,
                    values=match_wise_summary['Substitution impact'].value_counts(),
                    name='Substitution Impact',
                    textinfo='label+percent',
                    marker=dict(colors=['darkorange','blue']))
    fig.add_trace(trace1, row=1, col=1)
    fig.add_trace(trace2, row=1, col=2)
    fig.update_layout(
        title_text=f"Impact of Substitution for {team}",
        title_x=0.5,
        title_font=dict(size=24, family='Arial, sans-serif', color='darkblue'),
        showlegend=False
    )
    fig.show()

    return match_wise_summary
```

```
match_wise_summary = substitute_summary('Argentina')
```

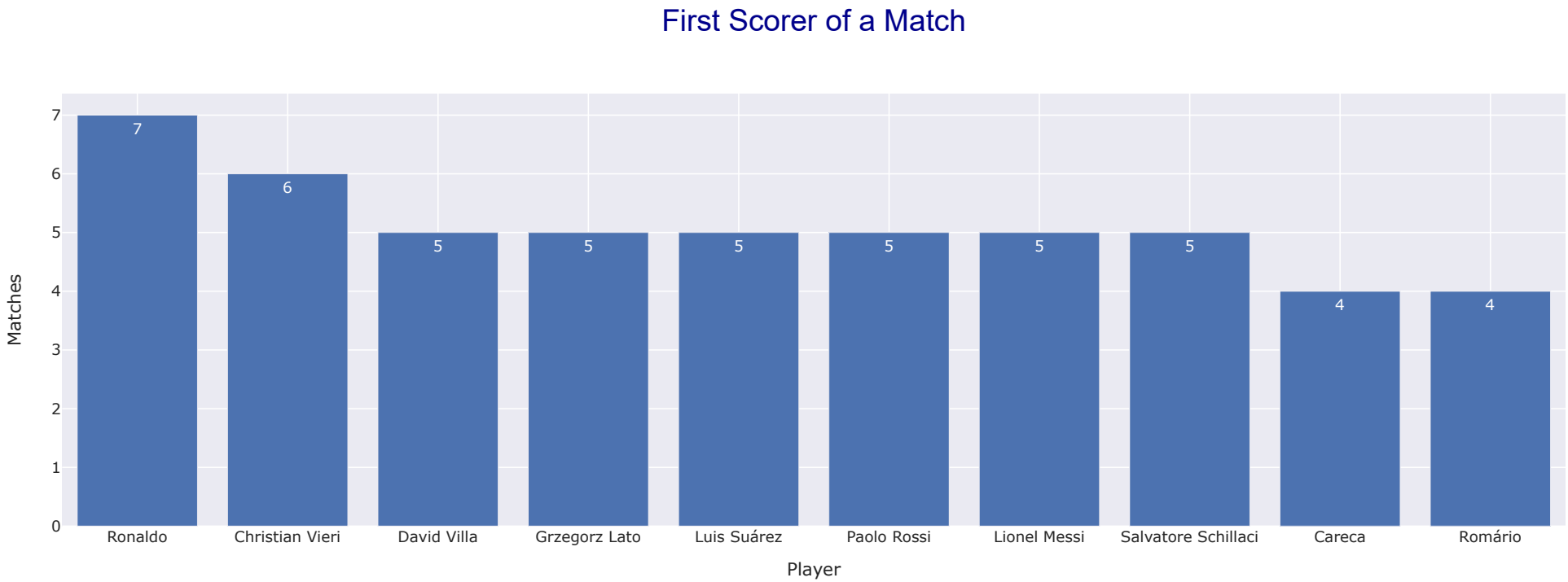


First Goal of a Match

```
def first_goal(row):
    home_goals = row['home_goal_details']
    away_goals = row['away_goal_details']
    if home_goals and (not away_goals or home_goals[0]['minute'] < away_goals[0]['minute']):
        return 'Home'
    elif away_goals:
        return 'Away'
    return 'None'
def first_goal_scorer(row):
    home_goals = row['home_goal_details']
    away_goals = row['away_goal_details']
    if home_goals and (not away_goals or home_goals[0]['minute'] < away_goals[0]['minute']):
        return home_goals[0]['scorer']
    elif away_goals:
        return away_goals[0]['scorer']
    return

matches['first_goal'] = matches.apply(first_goal, axis=1)
matches['first_scorer'] = matches.apply(first_goal_scorer,axis=1)
```

Players to score 1st goal of a match most times

```
first_scorers = matches['first_scorer'].value_counts().reset_index().head(10)

fig = px.bar(first_scorers, x='first_scorer', y='count', title='First Scorer of a Match',
             labels={'first_scorer': 'First Scorer', 'count': 'Number of Times Scored First'},
             text='count')

fig.update_layout(
    xaxis_title='Player',
    yaxis_title='Matches',
    title_x=0.5,
    title_font=dict(size=24, family='Arial, sans-serif', color='darkblue')
)

fig.show()
```



Conclusions:

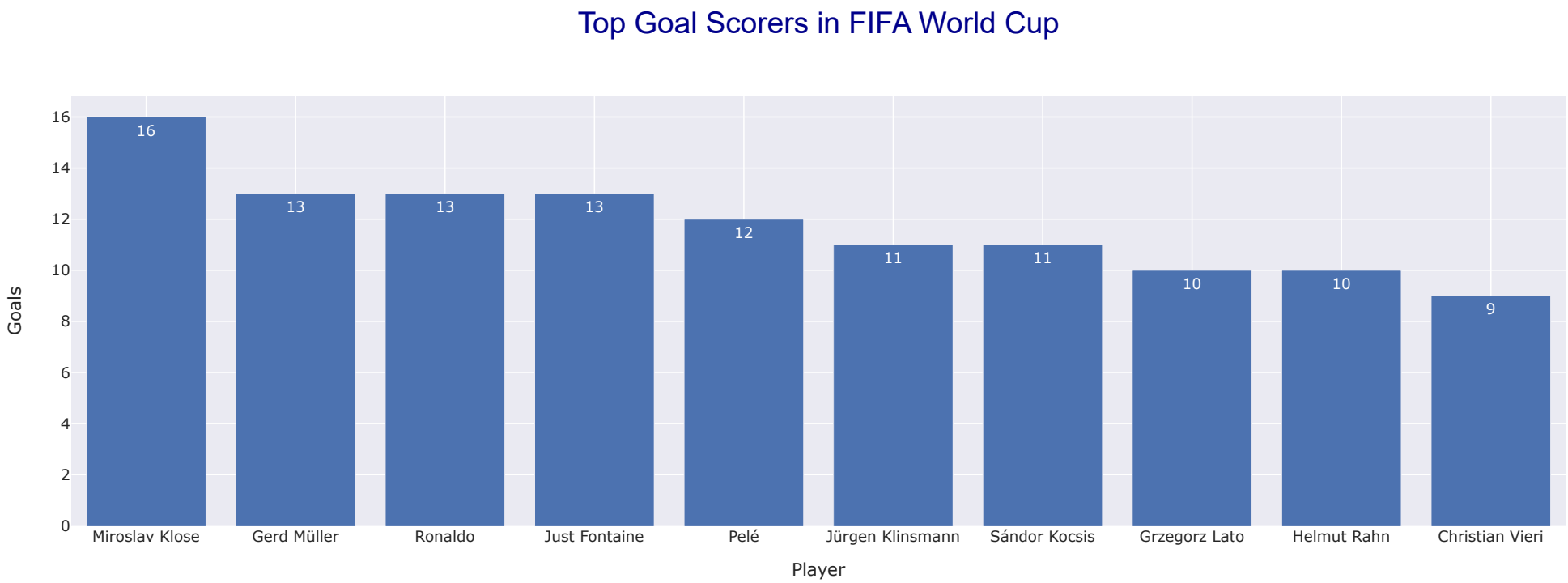
- Ronaldo has the highest number of first goals, with 7 instances. This indicates his significant impact and ability to score early in matches.
- Christian Vieri is next with 6 instances, showcasing his capability to make an early impact in matches.

Top Goal Scorers in FIFA World Cup

```
matches['home_goal_details'][0]
```

```
[{'minute': 36, 'scorer': 'Ángel Di María'},
 {'minute': 108, 'scorer': 'Lionel Messi'}]
```

```
home_goal_scorers = matches['home_goal_details'].explode().dropna().apply(lambda x: x['scorer'])
away_goal_scorers = matches['away_goal_details'].explode().dropna().apply(lambda x: x['scorer'])

all_goal_scorers = pd.concat([home_goal_scorers, away_goal_scorers])

goal_scorers_count = all_goal_scorers.value_counts().reset_index()
goal_scorers_count.columns = ['goal_scorer', 'count']
top_goal_scorers = goal_scorers_count.head(10)

fig = px.bar(top_goal_scorers, x='goal_scorer', y='count', title='Top Goal Scorers in FIFA World Cup',
             labels={'goal_scorer': 'Player', 'count': 'Goals'},
             text='count')
fig.update_layout(
    xaxis_title='Player',
    yaxis_title='Goals',
    title_x=0.5,
    title_font=dict(size=24, family='Arial, sans-serif', color='darkblue')
)
fig.show()
```



Performance When Team scored the first goal

```
def outcome_when_scoring_first(team):
    def scored_first(row):
        if (row['home_team']==team and row['first_goal']=='Home') or (row['away_team']==team and row['first_goal']=='Away'):
            return True
        return False
    def result(row):
        if (row['home_team']==team and row['result']=='Home Team Won') or (row['away_team']==team and row['result']=='Away Team Won'):
            return 'Win'
        elif row['result']=='Draw':
            return 'Draw'
        return 'Loss'

    team_matches = matches[(matches['home_team'] == team) | (matches['away_team'] == team)]
    team_matches['result'] = team_matches.apply(result, axis=1)
    team_matches['scored_first'] = team_matches.apply(scored_first, axis=1)

    outcomes = team_matches.groupby('scored_first')['result'].value_counts().reset_index()
    scored_first = outcomes[outcomes['scored_first']==True]
    not_scored_first = outcomes[outcomes['scored_first']==False]

    fig = make_subplots(rows=1, cols=2, subplot_titles=['Results when scoring first', 'Results when not scoring first'], specs=[[{'type': 'domain'}, {'type': 'domain'}]])
    trace1 = go.Pie(labels=scored_first['result'],
                    values=scored_first['count'],
                    name='Scored First',
                    textinfo='label+percent',
                    )
    trace2 = go.Pie(labels=not_scored_first['result'],
                    values=not_scored_first['count'],
                    name='Not Scored First',
                    textinfo='label+percent'
                    )
    fig.add_trace(trace1, row=1, col=1)
    fig.add_trace(trace2, row=1, col=2)
    fig.update_layout(
        title_text=f"Results when {team} scores first goal",
        title_x=0.5,
        title_font=dict(size=24, family='Arial, sans-serif', color='darkblue'),
        showlegend=False
    )
    fig.show()
```

```
outcome_when_scoring_first('Argentina')
```

Results when Argentina scores first goal