eda-fifa-wc22

April 27, 2025

1 EDA on FIFA WC 2022 Qatar

1.1 01- Importing Libraries

[21]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

1.2 02- Loading dataset

[22]:
```
df = pd.read_csv('Fifa_world_cup_matches.csv')
df.head(5)
```

[22]:
```
           team1         team2 possession team1 possession team2  \
0          QATAR       ECUADOR              42%              50%
1        ENGLAND          IRAN              72%              19%
2        SENEGAL   NETHERLANDS              44%              45%
3  UNITED STATES         WALES              51%              39%
4      ARGENTINA  SAUDI ARABIA              64%              24%

  possession in contest  number of goals team1  number of goals team2  \
0                    8%                      0                      2
1                    9%                      6                      2
2                   11%                      0                      2
3                   10%                      1                      1
4                   12%                      1                      2

          date     hour category  …  penalties scored team1  \
0  20 NOV 2022  17 : 00  Group A  …                       0
1  21 NOV 2022  14 : 00  Group B  …                       0
2  21 NOV 2022  17 : 00  Group A  …                       0
3  21 NOV 2022  20 : 00  Group B  …                       0
4  22 NOV 2022  11 : 00  Group C  …                       1

   penalties scored team2  goal preventions team1  goal preventions team2  \
0                       1                       6                       5
```

```
1                        1                        8                        13
2                        0                        9                        15
3                        1                        7                         7
4                        0                        4                        14

   own goals team1  own goals team2  forced turnovers team1  \
0                0                0                      52
1                0                0                      63
2                0                0                      63
3                0                0                      81
4                0                0                      65

   forced turnovers team2  defensive pressures applied team1  \
0                      72                                256
1                      72                                139
2                      73                                263
3                      72                                242
4                      80                                163

   defensive pressures applied team2
0                                279
1                                416
2                                251
3                                292
4                                361

[5 rows x 88 columns]
```

1.3 03- Applying EDA steps

[23]:
```
df.shape
```

[23]:
```
(64, 88)
```

[24]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 64 entries, 0 to 63
Data columns (total 88 columns):
 #   Column                                                 Non-Null Count
Dtype
---  ------                                                 --------------
-----
 0   team1                                                  64 non-null
object
 1   team2                                                  64 non-null
object
```

```
 2   possession team1                                 64 non-null
object
 3   possession team2                                 64 non-null
object
 4   possession in contest                            64 non-null
object
 5   number of goals team1                            64 non-null
int64
 6   number of goals team2                            64 non-null
int64
 7   date                                             64 non-null
object
 8   hour                                             64 non-null
object
 9   category                                         64 non-null
object
 10  total attempts team1                             64 non-null
int64
 11  total attempts team2                             64 non-null
int64
 12  conceded team1                                   64 non-null
int64
 13  conceded team2                                   64 non-null
int64
 14  goal inside the penalty area team1               64 non-null
int64
 15  goal inside the penalty area team2               64 non-null
int64
 16  goal outside the penalty area team1              64 non-null
int64
 17  goal outside the penalty area team2              64 non-null
int64
 18  assists team1                                    64 non-null
int64
 19  assists team2                                    64 non-null
int64
 20  on target attempts team1                         64 non-null
int64
 21  on target attempts team2                         64 non-null
int64
 22  off target attempts team1                        64 non-null
int64
 23  off target attempts team2                        64 non-null
int64
 24  attempts inside the penalty area team1           64 non-null
int64
 25  attempts inside the penalty area  team2          64 non-null
int64
```

```
 26  attempts outside the penalty area  team1              64 non-null
int64
 27  attempts outside the penalty area  team2              64 non-null
int64
 28  left channel team1                                    64 non-null
int64
 29  left channel team2                                    64 non-null
int64
 30  left inside channel team1                             64 non-null
int64
 31  left inside channel team2                             64 non-null
int64
 32  central channel team1                                 64 non-null
int64
 33  central channel team2                                 64 non-null
int64
 34  right inside channel team1                            64 non-null
int64
 35  right inside channel team2                            64 non-null
int64
 36  right channel team1                                   64 non-null
int64
 37  right channel team2                                   64 non-null
int64
 38  total offers to receive team1                         64 non-null
int64
 39  total offers to receive team2                         64 non-null
int64
 40  inbehind offers to receive team1                      64 non-null
int64
 41  inbehind offers to receive team2                      64 non-null
int64
 42  inbetween offers to receive team1                     64 non-null
int64
 43  inbetween offers to receive team2                     64 non-null
int64
 44  infront offers to receive team1                       64 non-null
int64
 45  infront offers to receive team2                       64 non-null
int64
 46  receptions between midfield and defensive lines team1  64 non-null
int64
 47  receptions between midfield and defensive lines team2  64 non-null
int64
 48  attempted line breaks team1                           64 non-null
int64
 49  attempted line breaks team2                           64 non-null
int64
```

```
 50  completed line breaksteam1                            64 non-null
int64
 51  completed line breaks team2                           64 non-null
int64
 52  attempted defensive line breaks team1                 64 non-null
int64
 53  attempted defensive line breaks team2                 64 non-null
int64
 54  completed defensive line breaksteam1                  64 non-null
int64
 55  completed defensive line breaks team2                 64 non-null
int64
 56  yellow cards team1                                    64 non-null
int64
 57  yellow cards team2                                    64 non-null
int64
 58  red cards team1                                       64 non-null
int64
 59  red cards team2                                       64 non-null
int64
 60  fouls against team1                                   64 non-null
int64
 61  fouls against team2                                   64 non-null
int64
 62  offsides team1                                        64 non-null
int64
 63  offsides team2                                        64 non-null
int64
 64  passes team1                                          64 non-null
int64
 65  passes team2                                          64 non-null
int64
 66  passes completed team1                                64 non-null
int64
 67  passes completed team2                                64 non-null
int64
 68  crosses team1                                         64 non-null
int64
 69  crosses team2                                         64 non-null
int64
 70  crosses completed team1                               64 non-null
int64
 71  crosses completed team2                               64 non-null
int64
 72  switches of play completed team1                      64 non-null
int64
 73  switches of play completed team2                      64 non-null
int64
```

```
 74  corners team1                                        64 non-null
int64
 75  corners team2                                        64 non-null
int64
 76  free kicks team1                                     64 non-null
int64
 77  free kicks team2                                     64 non-null
int64
 78  penalties scored team1                               64 non-null
int64
 79  penalties scored team2                               64 non-null
int64
 80  goal preventions team1                               64 non-null
int64
 81  goal preventions team2                               64 non-null
int64
 82  own goals team1                                      64 non-null
int64
 83  own goals team2                                      64 non-null
int64
 84  forced turnovers team1                               64 non-null
int64
 85  forced turnovers team2                               64 non-null
int64
 86  defensive pressures applied team1                    64 non-null
int64
 87  defensive pressures applied team2                    64 non-null
int64
dtypes: int64(80), object(8)
memory usage: 44.1+ KB
```

[25]:
```
df.describe()
```

[25]:
```
       number of goals team1  number of goals team2  total attempts team1  \
count              64.000000              64.000000             64.000000
mean                1.578125               1.109375             11.140625
std                 1.551289               1.055856              4.972519
min                 0.000000               0.000000              2.000000
25%                 0.000000               0.000000              8.000000
50%                 1.000000               1.000000             10.000000
75%                 2.000000               2.000000             14.000000
max                 7.000000               4.000000             25.000000

       total attempts team2  conceded team1  conceded team2  \
count             64.000000       64.000000       64.000000
mean              11.281250        1.109375        1.578125
std                5.807682        1.055856        1.551289
```

```
min                0.000000        0.000000        0.000000
25%                7.750000        0.000000        0.000000
50%               10.000000        1.000000        1.000000
75%               14.000000        2.000000        2.000000
max               32.000000        4.000000        7.000000

       goal inside the penalty area team1  goal inside the penalty area team2  \
count                           64.000000                           64.000000
mean                             1.468750                            0.984375
std                              1.563155                            0.999876
min                              0.000000                            0.000000
25%                              0.000000                            0.000000
50%                              1.000000                            1.000000
75%                              2.000000                            2.000000
max                              7.000000                            4.000000

       goal outside the penalty area team1  \
count                            64.000000
mean                              0.093750
std                               0.293785
min                               0.000000
25%                               0.000000
50%                               0.000000
75%                               0.000000
max                               1.000000

       goal outside the penalty area team2  …  penalties scored team1  \
count                            64.000000  …               64.000000
mean                              0.109375  …                0.140625
std                               0.314576  …                0.350382
min                               0.000000  …                0.000000
25%                               0.000000  …                0.000000
50%                               0.000000  …                0.000000
75%                               0.000000  …                0.000000
max                               1.000000  …                1.000000

       penalties scored team2  goal preventions team1  goal preventions team2  \
count               64.000000               64.000000               64.000000
mean                 0.125000               11.593750               11.359375
std                  0.377964                5.911299                4.990045
min                  0.000000                0.000000                2.000000
25%                  0.000000                7.750000                8.000000
50%                  0.000000               11.000000               10.000000
75%                  0.000000               14.000000               14.000000
max                  2.000000               32.000000               26.000000

       own goals team1  own goals team2  forced turnovers team1  \
count       64.000000        64.000000             64.000000
mean         0.015625         0.015625             71.968750
std          0.125000         0.125000             14.394629
min          0.000000         0.000000             38.000000
25%          0.000000         0.000000             63.000000
50%          0.000000         0.000000             71.000000
75%          0.000000         0.000000             83.500000
max          1.000000         1.000000            101.000000

       forced turnovers team2  defensive pressures applied team1  \
count               64.000000                          64.000000
mean                70.125000                         289.750000
std                 13.531269                          88.406888
min                 44.000000                         139.000000
25%                 60.250000                         229.000000
50%                 72.000000                         281.000000
75%                 79.000000                         328.000000
max                104.000000                         637.000000

       defensive pressures applied team2
count                          64.000000
mean                          293.265625
std                            80.916230
min                           141.000000
25%                           233.750000
50%                           292.500000
75%                           327.500000
max                           585.000000

[8 rows x 80 columns]
```

[26]:
```
df.isnull().sum()
```

[26]:
```
team1                                0
team2                                0
possession team1                     0
possession team2                     0
possession in contest                0
                                    ..
own goals team2                      0
forced turnovers team1               0
forced turnovers team2               0
defensive pressures applied team1    0
defensive pressures applied team2    0
Length: 88, dtype: int64
```

[27]:
```
df.describe().T
```

```
[27]:                                   count        mean        std    min  \
      number of goals team1              64.0    1.578125   1.551289    0.0
      number of goals team2              64.0    1.109375   1.055856    0.0
      total attempts team1               64.0   11.140625   4.972519    2.0
      total attempts team2               64.0   11.281250   5.807682    0.0
      conceded team1                     64.0    1.109375   1.055856    0.0
      …                                  …         …          …     …
      own goals team2                    64.0    0.015625   0.125000    0.0
      forced turnovers team1             64.0   71.968750  14.394629   38.0
      forced turnovers team2             64.0   70.125000  13.531269   44.0
      defensive pressures applied team1  64.0  289.750000  88.406888  139.0
      defensive pressures applied team2  64.0  293.265625  80.916230  141.0

                                            25%     50%     75%     max
      number of goals team1                0.00     1.0     2.0     7.0
      number of goals team2                0.00     1.0     2.0     4.0
      total attempts team1                 8.00    10.0    14.0    25.0
      total attempts team2                 7.75    10.0    14.0    32.0
      conceded team1                       0.00     1.0     2.0     4.0
      …                                   …     …     …     …
      own goals team2                      0.00     0.0     0.0     1.0
      forced turnovers team1              63.00    71.0    83.5   101.0
      forced turnovers team2              60.25    72.0    79.0   104.0
      defensive pressures applied team1  229.00   281.0   328.0   637.0
      defensive pressures applied team2  233.75   292.5   327.5   585.0

      [80 rows x 8 columns]
```

```
[28]: df["date"].head()
```

```
[28]: 0    20 NOV 2022
      1    21 NOV 2022
      2    21 NOV 2022
      3    21 NOV 2022
      4    22 NOV 2022
      Name: date, dtype: object
```

```
[29]: df["hour"].head()
```

```
[29]: 0    17 : 00
      1    14 : 00
      2    17 : 00
      3    20 : 00
      4    11 : 00
      Name: hour, dtype: object
```

```
[30]: # Possession to numeric
      df['possession team1'] = df['possession team1'].str.rstrip('%').astype(float)
      df['possession team2'] = df['possession team2'].str.rstrip('%').astype(float)
```

```
[31]: # New metrics
      df['total_goals'] = df['number of goals team1'] + df['number of goals team2']
      df['shot_accuracy_team1'] = df['number of goals team1'] / df['total attempts␣
       ↪team1']
      df['shot_accuracy_team2'] = df['number of goals team2'] / df['total attempts␣
       ↪team2']
```

```
[32]: df.head(5)
```

```
[32]:            team1          team2  possession team1  possession team2  \
      0          QATAR        ECUADOR              42.0              50.0
      1        ENGLAND           IRAN              72.0              19.0
      2        SENEGAL    NETHERLANDS              44.0              45.0
      3  UNITED STATES          WALES              51.0              39.0
      4      ARGENTINA   SAUDI ARABIA              64.0              24.0

        possession in contest  number of goals team1  number of goals team2  \
      0                    8%                      0                      2
      1                    9%                      6                      2
      2                   11%                      0                      2
      3                   10%                      1                      1
      4                   12%                      1                      2

               date     hour category  …  goal preventions team2  \
      0  20 NOV 2022  17 : 00  Group A  …                       5
      1  21 NOV 2022  14 : 00  Group B  …                      13
      2  21 NOV 2022  17 : 00  Group A  …                      15
      3  21 NOV 2022  20 : 00  Group B  …                       7
      4  22 NOV 2022  11 : 00  Group C  …                      14

         own goals team1  own goals team2  forced turnovers team1  \
      0                0                0                      52
      1                0                0                      63
      2                0                0                      63
      3                0                0                      81
      4                0                0                      65

         forced turnovers team2  defensive pressures applied team1  \
      0                      72                                256
      1                      72                                139
      2                      73                                263
      3                      72                                242
      4                      80                                163
```

```
   defensive pressures applied team2  total_goals  shot_accuracy_team1  \
0                                279            2             0.000000
1                                416            8             0.461538
2                                251            2             0.000000
3                                292            2             0.166667
4                                361            3             0.071429

   shot_accuracy_team2
0             0.333333
1             0.250000
2             0.222222
3             0.142857
4             0.666667

[5 rows x 91 columns]
```

1.4 04- Visualizations

1.4.1 Total Goals scored by Teams

[33]:
```python
import pandas as pd
import plotly.express as px

# Create new dataframes for team1 and team2 with their goals
team1_goals = df[["team1", "number of goals team1"]].rename(
    columns={"team1": "team", "number of goals team1": "goals"}
)
team2_goals = df[["team2", "number of goals team2"]].rename(
    columns={"team2": "team", "number of goals team2": "goals"}
)

# Combine the dataframes
all_teams_goals = pd.concat([team1_goals, team2_goals])

# Group by team and sum the goals
total_goals_by_team = all_teams_goals.groupby("team")["goals"].sum().
 ↪reset_index()

# Sort in descending order
total_goals_by_team = total_goals_by_team.sort_values(by="goals",␣
 ↪ascending=False)

# Create a bar chart
fig = px.bar(
    total_goals_by_team,
    x="team",
    y="goals",
    title="Total Goals Scored by Teams in FIFA World Cup",
    labels={"team": "Team", "goals": "Total Goals"},
    color="goals",
    color_continuous_scale="Viridis"
)

# Improve layout for better readability
fig.update_layout(
    xaxis=dict(
        tickangle=45,
        title_font=dict(size=14),
        tickfont=dict(size=12)
    ),
    yaxis=dict(
        title_font=dict(size=14),
        tickfont=dict(size=12)
    ),
    coloraxis_showscale=False,
    plot_bgcolor='white',
    hoverlabel=dict(bgcolor="white", font_size=12),
    margin=dict(b=100)
)

# Display the chart
fig.show()
```

1.4.2 Fewest Goals conceded by Teams

[34]:
```
import pandas as pd
import plotly.express as px

# Create new dataframes for team1 and team2 with their conceded goals
team1_conceded = df[["team1", "conceded team1"]].rename(
    columns={"team1": "team", "conceded team1": "conceded"}
)
team2_conceded = df[["team2", "conceded team2"]].rename(
    columns={"team2": "team", "conceded team2": "conceded"}
)

# Combine the dataframes
all_teams_conceded = pd.concat([team1_conceded, team2_conceded])

# Group by team and sum the conceded goals
total_conceded_by_team = all_teams_conceded.groupby("team")["conceded"].sum().
 ↪reset_index()
```

```
# Sort in ascending order (fewest goals conceded first)
total_conceded_by_team = total_conceded_by_team.sort_values(by="conceded",␣
 ↪ascending=True)

# Create a bar chart
fig = px.bar(
    total_conceded_by_team,
    x="team",
    y="conceded",
    title="Fewest Goals Conceded by Teams in FIFA World Cup",
    labels={"team": "Team", "conceded": "Goals Conceded"},
    color="conceded",
    color_continuous_scale="Viridis_r"  # Reversed color scale so lower is␣
 ↪better (darker)
)

# Improve layout for better readability
fig.update_layout(
    xaxis=dict(
        tickangle=45,
        title_font=dict(size=14),
        tickfont=dict(size=12)
    ),
    yaxis=dict(
        title_font=dict(size=14),
        tickfont=dict(size=12)
    ),
    coloraxis_showscale=False,
    plot_bgcolor='white',
    hoverlabel=dict(bgcolor="white", font_size=12),
    margin=dict(b=100)
)

# Display the chart
fig.show()
```

1.4.3 Possession vs Goals

[35]:
```
import pandas as pd
import plotly.express as px
import numpy as np
import plotly.graph_objects as go

# Process and combine team data
team_stats = (
    pd.concat([
        df[["team1", "possession team1", "number of goals team1"]].rename(
            columns={"team1": "team", "possession team1": "possession", "number␣
↪of goals team1": "goals"}
        ),
        df[["team2", "possession team2", "number of goals team2"]].rename(
            columns={"team2": "team", "possession team2": "possession", "number␣
↪of goals team2": "goals"}
        )
    ])
    .assign(
        # First check if possession is string before stripping %
        possession=lambda x: x["possession"].apply(lambda val: float(str(val).
↪rstrip('%')) if isinstance(val, str) else float(val)),
        goals=lambda x: pd.to_numeric(x["goals"])
    )
    .groupby("team")
    .agg(
        avg_possession=("possession", "mean"),
        total_goals=("goals", "sum"),
        matches_played=("goals", "count")
    )
    .reset_index()
)

# Calculate correlation and trendline
x, y = team_stats["avg_possession"], team_stats["total_goals"]
corr = np.corrcoef(x, y)[0, 1]
z = np.poly1d(np.polyfit(x, y, 1))

# Create visualization
fig = px.scatter(
    team_stats,
    x="avg_possession",
    y="total_goals",
    text="team",
    size="matches_played",
    title=f"Possession vs Goals (Correlation: {corr:.2f})",
    labels={
        "avg_possession": "Avg Possession (%)",
        "total_goals": "Total Goals",
        "matches_played": "Matches Played"
    },
    color="total_goals",
    color_continuous_scale="Viridis"
).update_traces(
    textposition='top center',
    marker=dict(opacity=0.8)
).add_traces(
    go.Scatter(
        x=np.linspace(x.min(), x.max(), 100),
        y=z(np.linspace(x.min(), x.max(), 100)),
        mode='lines',
        name='Trend',
        line=dict(color='red', dash='dash'),
        opacity=0.7
    )
).update_layout(
    xaxis=dict(title_font=dict(size=14)),
    yaxis=dict(title_font=dict(size=14)),
    plot_bgcolor='white',
    xaxis_range=[30, 70]
)

fig.show()
```

1.4.4 Shot Conversion Efficiency

[36]:
```
# Process and analyze data
team_stats = (
    pd.concat([
        df[["team1", "total attempts team1", "number of goals team1"]]
            .rename(columns={"team1": "team", "total attempts team1": "shots",␣
 ↪"number of goals team1": "goals"}),
        df[["team2", "total attempts team2", "number of goals team2"]]
            .rename(columns={"team2": "team", "total attempts team2": "shots",␣
 ↪"number of goals team2": "goals"})
    ])
    .assign(
        shots=lambda x: pd.to_numeric(x["shots"]),
        goals=lambda x: pd.to_numeric(x["goals"])
    )
    .groupby("team")
    .agg(total_shots=("shots", "sum"), total_goals=("goals", "sum"),␣
 ↪matches_played=("shots", "count"))
    .query("total_shots >= 10")  # Filter minimum shots
    .assign(conversion_rate=lambda x: (x["total_goals"]/x["total_shots"]*100).
 ↪round(2))
    .sort_values("conversion_rate", ascending=False)
    .reset_index()
)

# Create visualization
fig = px.bar(
    team_stats,
    x="team",
    y="conversion_rate",
    title="Shot Conversion Efficiency (Goals per 100 Shots)",
    labels={"team": "Team", "conversion_rate": "Conversion Rate (%)"},
    color="conversion_rate",
    color_continuous_scale="Viridis",
    hover_data=["total_goals", "total_shots", "matches_played"]
).update_traces(
    texttemplate='%{y:.1f}%',
    textposition='outside'
).update_layout(
    xaxis=dict(tickangle=45, title_font=dict(size=14), tickfont=dict(size=12)),
    yaxis=dict(title_font=dict(size=14), tickfont=dict(size=12), range=[0,␣
 ↪team_stats["conversion_rate"].max()*1.1]),
    coloraxis_showscale=False,
    plot_bgcolor='white',
    hoverlabel=dict(bgcolor="white", font_size=12),
    margin=dict(b=100)
)

# Add average line
avg = team_stats["conversion_rate"].mean()
fig.add_shape(
    type="line",
    x0=-0.5, y0=avg, x1=len(team_stats)-0.5, y1=avg,
    line=dict(color="red", width=2, dash="dash")
).add_annotation(
    x=len(team_stats)-1, y=avg,
    text=f"Average: {avg:.2f}%",
    showarrow=False, yshift=10, font=dict(color="red")
)

fig.show()
```

1.4.5 Defensive Pressures vs Goals Conceded

[37]:
```
import plotly.express as px
import plotly.graph_objects as go

# Process and analyze defensive data
team_stats = (
    pd.concat([
        df[["team1", "defensive pressures applied team1", "conceded team1"]]
            .rename(columns={
                "team1": "team",
                "defensive pressures applied team1": "pressures",
                "conceded team1": "goals_conceded"
            }),
        df[["team2", "defensive pressures applied team2", "conceded team2"]]
            .rename(columns={
                "team2": "team",
                "defensive pressures applied team2": "pressures",
                "conceded team2": "goals_conceded"
            })
    ])
    .assign(
        pressures=lambda x: x["pressures"].apply(
            lambda v: float(str(v).rstrip('%')) if isinstance(v, str) or '%' in␣
↪str(v) else float(v)
        ),
        goals_conceded=lambda x: pd.to_numeric(x["goals_conceded"],␣
↪errors='coerce')
    )
    .dropna()
    .groupby("team")
    .agg(
        total_pressures=("pressures", "sum"),
        total_conceded=("goals_conceded", "sum"),
        matches=("goals_conceded", "count")
    )
    .query("matches > 0")
    .assign(
        avg_pressures=lambda x: (x["total_pressures"]/x["matches"]).round(1),
        avg_conceded=lambda x: (x["total_conceded"]/x["matches"]).round(2)
    )
    .reset_index()
)

# Calculate correlation and trendline
x, y = team_stats["avg_pressures"], team_stats["avg_conceded"]
corr = np.corrcoef(x, y)[0, 1]
trend = np.poly1d(np.polyfit(x, y, 1))

# Create visualization
fig = px.scatter(
    team_stats,
    x="avg_pressures",
    y="avg_conceded",
    text="team",
    size="matches",
    title=f"Defensive Pressures vs Goals Conceded (Correlation: {corr:.2f})",
    labels={
        "avg_pressures": "Avg Defensive Pressures/Match",
        "avg_conceded": "Avg Goals Conceded/Match",
        "matches": "Matches Played"
    },
    color="avg_conceded",
    color_continuous_scale="RdYlGn_r",
    width=1000,
    height=700
).update_traces(
    textposition='top center',
    marker=dict(opacity=0.8, sizemode='diameter', sizeref=0.1),
    textfont=dict(size=12)
).update_layout(
    font=dict(size=12),
    xaxis=dict(title_font=dict(size=14)),
    yaxis=dict(title_font=dict(size=14)),
    plot_bgcolor='white'
)

# Add trendline
fig.add_trace(
    go.Scatter(
        x=np.linspace(x.min(), x.max(), 100),
        y=trend(np.linspace(x.min(), x.max(), 100)),
        mode='lines',
        name='Trend',
        line=dict(color='red', dash='dash', width=2)
    )
)

# Add quadrant analysis
avg_x, avg_y = x.mean(), y.mean()
fig.update_layout(
    shapes=[
        dict(type="line", x0=avg_x, y0=y.min(), x1=avg_x, y1=y.max(),␣
 ↪line=dict(color="gray", dash="dot")),
        dict(type="line", x0=x.min(), y0=avg_y, x1=x.max(), y1=avg_y,␣
 ↪line=dict(color="gray", dash="dot"))
    ],
    annotations=[
        dict(x=x.max()*0.25, y=y.max()*0.25, text="High Efficiency",␣
 ↪showarrow=False, font=dict(color="green")),
        dict(x=x.max()*0.75, y=y.max()*0.25, text="Active Defense",␣
 ↪showarrow=False, font=dict(color="darkgreen")),
        dict(x=x.max()*0.25, y=y.max()*0.75, text="Passive Defense",␣
 ↪showarrow=False, font=dict(color="darkred")),
        dict(x=x.max()*0.75, y=y.max()*0.75, text="Ineffective Defense",␣
 ↪showarrow=False, font=dict(color="red"))
```

```
    ]
)

fig.show()
```

1.4.6 Teams Performance: Group vs Knockout Stage

[38]:
```
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# Process data
def process_data(matches_df, stage_name):
    return (
        pd.concat([
            matches_df[["team1", "number of goals team1", "conceded team1",␣
 ↪"possession team1",
                        "total attempts team1", "passes team1", "passes completed␣
 ↪team1"]]
                .rename(columns={"team1": "team", "number of goals team1":␣
 ↪"goals", "conceded team1": "conceded",
                                 "possession team1": "possession", "total␣
 ↪attempts team1": "attempts",
                                 "passes team1": "passes", "passes completed␣
 ↪team1": "completed_passes"}),
            matches_df[["team2", "number of goals team2", "conceded team2",␣
 ↪"possession team2",
                        "total attempts team2", "passes team2", "passes completed␣
 ↪team2"]]
                .rename(columns={"team2": "team", "number of goals team2":␣
 ↪"goals", "conceded team2": "conceded",
                                 "possession team2": "possession", "total␣
 ↪attempts team2": "attempts",
                                 "passes team2": "passes", "passes completed␣
 ↪team2": "completed_passes"})
        ])
        .assign(**{col: lambda x: pd.to_numeric(x[col]) for col in ["goals",␣
 ↪"conceded", "possession", "attempts", "passes", "completed_passes"]})
        .groupby("team")
        .agg(
            total_goals=("goals", "sum"),
            total_conceded=("conceded", "sum"),
            avg_possession=("possession", "mean"),
            total_attempts=("attempts", "sum"),
            total_passes=("passes", "sum"),
            completed_passes=("completed_passes", "sum"),
            matches_played=("goals", "count")
        )
        .assign(
            goals_per_match=lambda x: (x["total_goals"] / x["matches_played"]).
 ↪round(2),
            conceded_per_match=lambda x: (x["total_conceded"] /␣
 ↪x["matches_played"]).round(2),
            shot_conversion=lambda x: (x["total_goals"] / x["total_attempts"] *␣
 ↪100).round(1),
            passing_accuracy=lambda x: (x["completed_passes"] /␣
 ↪x["total_passes"] * 100).round(1),
            avg_possession_pct=lambda x: (x["avg_possession"] * 100).round(1),
            stage=stage_name
        )
        .reset_index()
    )

# Process stages
df["category"] = df["category"].astype(str)
group_stats = process_data(df[df["category"].str.contains("Group")], "Group␣
 ↪Stage")
knockout_stats = process_data(df[~df["category"].str.contains("Group")],␣
 ↪"Knockout Stage")
knockout_teams = set(knockout_stats["team"])
group_knockout_stats = group_stats[group_stats["team"].isin(knockout_teams)]
comparison_df = pd.concat([group_knockout_stats, knockout_stats])

# Create visualization
fig = make_subplots(
    rows=2, cols=2,
    subplot_titles=("Goals Per Match", "Goals Conceded Per Match",
                   "Shot Conversion Rate (%)", "Passing Accuracy (%)"),
    specs=[[{}, {}], [{}, {}]],
    vertical_spacing=0.15
)

metrics = [
    {"name": "goals_per_match", "row": 1, "col": 1, "ascending": False},
    {"name": "conceded_per_match", "row": 1, "col": 2, "ascending": True},
    {"name": "shot_conversion", "row": 2, "col": 1, "ascending": False},
    {"name": "passing_accuracy", "row": 2, "col": 2, "ascending": False}
]

for metric in metrics:
    top_teams = knockout_stats.sort_values(metric["name"],␣
↪ascending=metric["ascending"])["team"].head(8)
    plot_data = comparison_df[comparison_df["team"].isin(top_teams)]

    for stage, color in [("Group Stage", "royalblue"), ("Knockout Stage",␣
↪"darkred")]:
        stage_data = plot_data[plot_data["stage"] == stage].
↪sort_values(metric["name"], ascending=metric["ascending"])
        fig.add_trace(
            go.Bar(
                x=stage_data["team"],
                y=stage_data[metric["name"]],
                name=stage,
                marker_color=color,
                text=stage_data[metric["name"]].round(1),
                textposition="outside",
                showlegend=(metric["row"] == 1 and metric["col"] == 1)
            ),
            row=metric["row"],
            col=metric["col"]
        )

fig.update_layout(
    title_text="Team Performance: Group Stage vs. Knockout Stage",
    barmode="group",
    height=700,
    width=1000,
    plot_bgcolor="white",
    legend=dict(orientation="h", yanchor="bottom", y=1.02, xanchor="center",␣
↪x=0.5),
    margin=dict(t=100)
).update_xaxes(tickangle=45)

fig.show()
```

1.4.7 Possession vs Goals

[40]:
```
import pandas as pd
import plotly.express as px
import numpy as np
import plotly.graph_objects as go

# Process and combine team data
team_stats = (
    pd.concat([
        df[["team1", "possession team1", "number of goals team1"]].rename(
            columns={"team1": "team", "possession team1": "possession", "number␣
↪of goals team1": "goals"}
        ),
        df[["team2", "possession team2", "number of goals team2"]].rename(
            columns={"team2": "team", "possession team2": "possession", "number␣
↪of goals team2": "goals"}
        )
    ])
    .assign(
        # First check if possession is string before stripping %
        possession=lambda x: x["possession"].apply(lambda val: float(str(val).
↪rstrip('%')) if isinstance(val, str) else float(val)),
        goals=lambda x: pd.to_numeric(x["goals"])
    )
    .groupby("team")
    .agg(
        avg_possession=("possession", "mean"),
        total_goals=("goals", "sum"),
        matches_played=("goals", "count")
    )
    .reset_index()
)

# Calculate correlation and trendline
x, y = team_stats["avg_possession"], team_stats["total_goals"]
corr = np.corrcoef(x, y)[0, 1]
z = np.poly1d(np.polyfit(x, y, 1))

# Create visualization
fig = px.scatter(
    team_stats,
    x="avg_possession",
    y="total_goals",
    text="team",
    size="matches_played",
    title=f"Possession vs Goals (Correlation: {corr:.2f})",
    labels={
        "avg_possession": "Avg Possession (%)",
        "total_goals": "Total Goals",
        "matches_played": "Matches Played"
    },
    color="total_goals",
    color_continuous_scale="Viridis"
).update_traces(
    textposition='top center',
    marker=dict(opacity=0.8)
).add_traces(
    go.Scatter(
        x=np.linspace(x.min(), x.max(), 100),
        y=z(np.linspace(x.min(), x.max(), 100)),
        mode='lines',
        name='Trend',
        line=dict(color='red', dash='dash'),
        opacity=0.7
    )
).update_layout(
    xaxis=dict(title_font=dict(size=14)),
    yaxis=dict(title_font=dict(size=14)),
    plot_bgcolor='white',
    xaxis_range=[30, 70]
)

fig.show()
```

1.4.8 Set Pieces Effectiveness

[41]:
```
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# Process team data in one pipeline
team_stats = (
    pd.concat([
        df[["team1", "corners team1", "free kicks team1", "number of goals␣
 ↪team1",
            "crosses team1", "crosses completed team1"]]
            .rename(columns={
                "team1": "team",
                "corners team1": "corners",
                "free kicks team1": "free_kicks",
                "number of goals team1": "goals",
                "crosses team1": "crosses",
                "crosses completed team1": "completed_crosses"
            }),
        df[["team2", "corners team2", "free kicks team2", "number of goals␣
 ↪team2",
            "crosses team2", "crosses completed team2"]]
            .rename(columns={
                "team2": "team",
                "corners team2": "corners",
                "free kicks team2": "free_kicks",
                "number of goals team2": "goals",
                "crosses team2": "crosses",
                "crosses completed team2": "completed_crosses"
            })
    ])
    .assign(**{col: lambda x: pd.to_numeric(x[col])
            for col in ["corners", "free_kicks", "goals", "crosses",␣
↪"completed_crosses"]})
    .groupby("team")
    .agg(
        total_corners=("corners", "sum"),
        total_free_kicks=("free_kicks", "sum"),
        total_goals=("goals", "sum"),
        total_crosses=("crosses", "sum"),
        completed_crosses=("completed_crosses", "sum"),
        matches_played=("corners", "count")
    )
    .assign(
        corners_per_match=lambda x: (x["total_corners"]/x["matches_played"]).
↪round(1),
        free_kicks_per_match=lambda x: (x["total_free_kicks"]/
↪x["matches_played"]).round(1),
        goals_per_match=lambda x: (x["total_goals"]/x["matches_played"]).
↪round(1),
        cross_completion_rate=lambda x: (x["completed_crosses"]/
↪x["total_crosses"]*100).round(1),
        set_pieces_per_goal=lambda x:␣
↪((x["total_corners"]+x["total_free_kicks"])/x["total_goals"]).round(1),
        set_piece_efficiency=lambda x: (100/
↪((x["total_corners"]+x["total_free_kicks"])/x["total_goals"])).round(1)
    )
    .reset_index()
)

# Create subplots
fig = make_subplots(
    rows=2, cols=1,
    subplot_titles=("Set Piece Efficiency (%)", "Set Pieces Per Goal"),
    vertical_spacing=0.25
)

# Add traces
for i, (metric, color, title) in enumerate([
    ("set_piece_efficiency", px.colors.sequential.Viridis, "Efficiency"),
    ("set_pieces_per_goal", px.colors.sequential.Viridis[::-1], "Pieces/Goal")
]):
    data = team_stats.sort_values(metric, ascending=i==1).head(16)
    fig.add_trace(
        go.Bar(
            x=data["team"],
            y=data[metric],
            text=data[metric].apply(lambda x: f"{x:.1f}{'%' if i==0 else ''}"),
            textposition="outside",
            marker_color=color,
            name=title,
            customdata=data[["corners_per_match", "free_kicks_per_match"]],
            hovertemplate="<b>%{x}</b><br>%{y:.1f}" +
                        ("%<br>Corners/match: %{customdata[0]:.1f}<br>Free␣
↪kicks/match: %{customdata[1]:.1f}" if i==0 else
                        "<br>Corners: %{customdata[0]:.1f}<br>Free kicks:␣
↪%{customdata[1]:.1f}")
        ),
        row=i+1, col=1
    )

# Update layout
fig.update_layout(
    title_text="Set Piece Effectiveness Analysis",
    height=800,
    width=1000,
    showlegend=False,
    plot_bgcolor='white',
    xaxis1=dict(tickangle=45),
    xaxis2=dict(tickangle=45),
    yaxis1=dict(title="Efficiency (%)", range=[0,␣
↪team_stats["set_piece_efficiency"].max()*1.1]),
    yaxis2=dict(title="Pieces/Goal", range=[0,␣
↪team_stats["set_pieces_per_goal"].median()*2])
)

# Create scatter plot
fig2 = px.scatter(
    team_stats,
    x="corners_per_match",
    y="free_kicks_per_match",
    size="goals_per_match",
    color="cross_completion_rate",
    hover_name="team",
    title="Set Piece Usage Analysis",
    labels={
        "corners_per_match": "Corners/Match",
        "free_kicks_per_match": "Free Kicks/Match",
        "goals_per_match": "Goals/Match",
        "cross_completion_rate": "Cross Completion %"
    },
    color_continuous_scale="Viridis"
).update_traces(
    textposition='top center',
    marker=dict(opacity=0.8)
).update_layout(
    height=600,
    width=800,
    plot_bgcolor='white'
)

fig.show()
fig2.show()
```

1.4.9 Goals Scored Per Tournament Stage

```
[51]: total_goals = (
          pd.concat([
              df['number of goals team1'],
              df['number of goals team2']
          ]).sum()
      )
      print(f"Total goals in tournament: {total_goals}")
```

```
Total goals in tournament: 172
```

```
[49]: import plotly.express as px
      import pandas as pd

      # Calculate total goals per match
      df['total_goals'] = df['number of goals team1'] + df['number of goals team2']

      # Group by tournament category/stage and calculate total goals
      goals_by_stage = df.groupby('category').agg({
          'total_goals': 'sum',
          'team1': 'count'  # Count number of matches
      }).reset_index()

      goals_by_stage.rename(columns={'team1': 'matches'}, inplace=True)
      goals_by_stage['avg_goals_per_match'] = round(goals_by_stage['total_goals'] /␣
        ↪goals_by_stage['matches'], 2)

      # Sort by tournament progression
      stage_order = ['Group A', 'Group B', 'Group C', 'Group D', 'Group E', 'Group F',
                     'Group G', 'Group H', 'Round of 16', 'Quarter-final',␣
        ↪'Semi-final',
                     'Third place', 'Final']
```

```
goals_by_stage['category'] = pd.Categorical(goals_by_stage['category'],␣
 ↪categories=stage_order, ordered=True)
goals_by_stage = goals_by_stage.sort_values('category')

# Create visualization
fig = px.bar(goals_by_stage, x='category', y='total_goals',
             title='Total Goals by Tournament Stage',
             labels={'category': 'Tournament Stage', 'total_goals': 'Total␣
 ↪Goals'},
             color='avg_goals_per_match',
             color_continuous_scale='RdBu',
             text='total_goals')

# Improve layout
fig.update_layout(
    xaxis=dict(
        tickangle=45,
        title_font=dict(size=14),
        tickfont=dict(size=12),
    ),
    yaxis=dict(
        title_font=dict(size=14),
        tickfont=dict(size=12),
    ),
    plot_bgcolor='white',
    coloraxis_colorbar=dict(
        title='Avg Goals per Match',
    )
)

fig.update_traces(textposition='outside')

fig.show()
```

1.4.10 Total Goals scored by Teams

[46]:
```
import plotly.express as px
import pandas as pd

# Calculate total goals by team
team_goals = (
    pd.concat([
        df[["team1", "number of goals team1"]].rename(columns={"team1": "team",␣
 ↪"number of goals team1": "goals"}),
        df[["team2", "number of goals team2"]].rename(columns={"team2": "team",␣
 ↪"number of goals team2": "goals"})
```

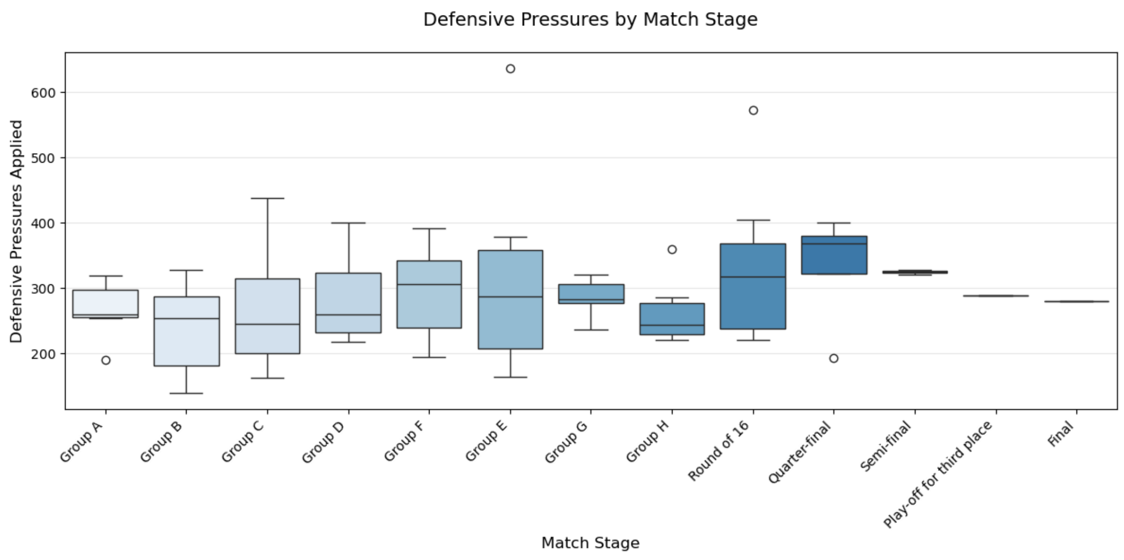
```
    ])
    .groupby("team")["goals"]
    .sum()
    .reset_index()
    .sort_values("goals", ascending=False)
)

# Create interactive pie chart
fig = px.pie(team_goals,
             values='goals',
             names='team',
             title='Percentage of Total Goals Scored by Each Team',
             hover_data=['goals'],
             labels={'goals': 'Total Goals'},
             color_discrete_sequence=px.colors.qualitative.Plotly)

# Enhance hover template
fig.update_traces(
    hovertemplate="<b>%{label}</b><br>Goals: %{value}<br>Percent: %{percent}",
    textinfo='percent+label',
    textposition='inside',
    marker=dict(line=dict(color='white', width=1)))

# Optional: Make it a donut chart
fig.update_layout(
    uniformtext_minsize=12,
    uniformtext_mode='hide',
    showlegend=True,
    annotations=[dict(text='Goals', x=0.5, y=0.5, font_size=20,␣
 ↪showarrow=False)]
)

fig.show()
```

1.4.11 Defensive Pressures by Match Stage

```
[ ]: plt.figure(figsize=(12, 6))  # Set figure size (width, height)

     # Create boxplot with adjusted parameters
     ax = sns.boxplot(
         x='category',
         y='defensive pressures applied team1',
         data=df,
         palette="Blues"  # Optional: adds color
     )

     # Rotate x-axis labels for better readability
```

```
plt.xticks(rotation=45, ha='right')  # Rotate 45 degrees and right-align

# Adjust title and labels
plt.title("Defensive Pressures by Match Stage", fontsize=14, pad=20)
plt.xlabel("Match Stage", fontsize=12)
plt.ylabel("Defensive Pressures Applied", fontsize=12)

# Adjust layout to prevent cutting off labels
plt.tight_layout()

# Optional: Add grid for better readability
plt.grid(axis='y', alpha=0.3)

# Show plot
plt.show()
```

```
C:\Users\Yawar Ali\AppData\Local\Temp\ipykernel_620\1773971818.py:6:
FutureWarning:

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

  ax = sns.boxplot(
```



1.4.12 Teams Crossing Performance

[63]:
```python
import pandas as pd
import plotly.express as px

# Calculate crossing stats for each team
teams = set(df['team1']).union(set(df['team2']))
team_stats = []

for team in teams:
    team_matches = df[(df['team1'] == team) | (df['team2'] == team)]
    if len(team_matches) == 0:
        continue

    is_team1 = (team_matches['team1'] == team)
    wins = sum(
        (team_matches['number of goals team1'] > team_matches['number of goals␣
↪team2']) & is_team1 |
        (team_matches['number of goals team2'] > team_matches['number of goals␣
↪team1']) & ~is_team1
    )

    crosses_attempted = (
        team_matches.loc[is_team1, 'crosses team1'].sum() +
        team_matches.loc[~is_team1, 'crosses team2'].sum()
    )
    crosses_completed = (
        team_matches.loc[is_team1, 'crosses completed team1'].sum() +
        team_matches.loc[~is_team1, 'crosses completed team2'].sum()
    )

    team_stats.append({
        'team': team,
        'matches': len(team_matches),
        'crosses_per_match': round(crosses_attempted/len(team_matches), 2),
        'win_rate': round(wins/len(team_matches)*100, 1),
        'completion_rate': round(crosses_completed/crosses_attempted*100, 1) if␣
↪crosses_attempted > 0 else 0
    })

crossing_df = pd.DataFrame(team_stats)

# Create the scatter plot
fig = px.scatter(
    crossing_df,
    x='crosses_per_match',
    y='win_rate',
    color='completion_rate',
    size='matches',
    hover_name='team',
    color_continuous_scale='Blues',
    labels={
        'crosses_per_match': 'Crosses per Match',
        'win_rate': 'Win Rate (%)',
        'completion_rate': 'Completion Rate (%)',
        'matches': 'Matches Played'
    },
    title='Crossing Effectiveness vs. Win Rate: 2022 FIFA World Cup'
)

# Customize the layout
fig.update_layout(
    plot_bgcolor='white',
    hovermode='closest',
    height=600,
    xaxis=dict(range=[5, 25]),  # Adjust based on your data
    yaxis=dict(range=[0, 100])
)

# Improve hover template
fig.update_traces(
    hovertemplate=(
        "<b>%{hovertext}</b><br>"
        "Crosses/Match: %{x:.1f}<br>"
        "Win Rate: %{y:.1f}%<br>"
        "Completion: %{marker.color:.1f}%<br>"
        "Matches: %{marker.size:.0f}"
    ),
    marker=dict(opacity=0.8, line=dict(width=1, color='DarkSlateGrey'))
)

fig.show()
```

1.4.13 Team Crossing Performance Analysis

[67]:
```
import pandas as pd
import plotly.graph_objects as go
from plotly.subplots import make_subplots

simplified_stages = ['Group Stage', 'Round of 16', 'Quarter-final',␣
 ↪'Semi-final', 'Third place', 'Final']
df['simplified_stage'] = df['category'].apply(lambda x: 'Group Stage' if␣
 ↪'Group' in x else x)
```

```
# 1. Calculate team-level crossing statistics
team_stats = []
for team in pd.unique(pd.concat([df['team1'], df['team2']])):
    team_matches = df[(df['team1'] == team) | (df['team2'] == team)]
    if len(team_matches) == 0:
        continue

    is_team1 = (team_matches['team1'] == team)
    crosses_attempted = (team_matches.loc[is_team1, 'crosses team1'].sum() +
                        team_matches.loc[~is_team1, 'crosses team2'].sum())
    crosses_completed = (team_matches.loc[is_team1, 'crosses completed team1'].
 ↪sum() +
                        team_matches.loc[~is_team1, 'crosses completed team2'].
 ↪sum())

    team_stats.append({
        'team': team,
        'matches': len(team_matches),
        'avg_crosses': crosses_attempted / len(team_matches),
        'completion_rate': (crosses_completed / crosses_attempted * 100) if␣
 ↪crosses_attempted > 0 else 0,
        'stage_reached': team_matches['simplified_stage'].max()  # Highest␣
 ↪stage reached
    })

team_df = pd.DataFrame(team_stats)

# 2. Calculate stage-level statistics (original code)
stage_stats = []
for stage in simplified_stages:
    stage_matches = df[df['simplified_stage'] == stage]
    if len(stage_matches) == 0:
        continue

    team_count = len(stage_matches) * 2
    stage_stats.append({
        'stage': stage,
        'avg_crosses': (stage_matches['crosses team1'].sum() +␣
 ↪stage_matches['crosses team2'].sum()) / team_count,
        'avg_completed': (stage_matches['crosses completed team1'].sum() +␣
 ↪stage_matches['crosses completed team2'].sum()) / team_count,
        'completion_rate': (stage_matches['crosses completed team1'].sum() +␣
 ↪stage_matches['crosses completed team2'].sum()) /
                          (stage_matches['crosses team1'].sum() +␣
 ↪stage_matches['crosses team2'].sum()) * 100,
        'avg_goals': (stage_matches['number of goals team1'].sum() +␣
 ↪stage_matches['number of goals team2'].sum()) / team_count
    })

stage_df = pd.DataFrame(stage_stats).sort_values('stage', key=lambda x: x.
 ↪map({v:i for i,v in enumerate(simplified_stages)}))

# Create dashboard with subplots
fig = make_subplots(
    rows=2, cols=1,
    subplot_titles=(
        "Crossing Performance by Tournament Stage",
        "Team Crossing Efficiency by Stage Reached"
    ),
    vertical_spacing=0.15,
    specs=[[{"type": "bar"}], [{"type": "scatter"}]]
)

# 1. Stage-level plot (top)
for col, name, color in [('avg_crosses', 'Avg Crosses', 'royalblue'),
                         ('avg_completed', 'Avg Completed', 'darkblue')]:
    fig.add_trace(go.Bar(
        x=stage_df['stage'], y=stage_df[col], name=name,
        marker_color=color, opacity=0.8, showlegend=True
    ), row=1, col=1)

for col, name, color, dash in [('completion_rate', 'Completion Rate', 'red',␣
 ↪None),
                              ('avg_goals', 'Avg Goals', 'green', 'dash')]:
    fig.add_trace(go.Scatter(
        x=stage_df['stage'], y=stage_df[col], name=name,
        mode='lines+markers', yaxis='y2',
        line=dict(color=color, width=3, dash=dash),
        marker=dict(size=10), showlegend=True
    ), row=1, col=1)

# 2. Team-level plot (bottom)
fig.add_trace(go.Scatter(
    x=team_df['avg_crosses'],
    y=team_df['completion_rate'],
    mode='markers',
    marker=dict(
        size=team_df['matches']*3,
        color=team_df['stage_reached'].map({v:i for i,v in␣
 ↪enumerate(simplified_stages)}),
        colorscale='Viridis',
        showscale=True,
        colorbar=dict(title='Stage Reached')
    ),
    text=team_df['team'],
    hoverinfo='text',
    hovertext=team_df.apply(
        lambda r: f"<b>{r['team']}</b><br>"
                  f"Stage: {r['stage_reached']}<br>"
                  f"Matches: {r['matches']}<br>"
                  f"Crosses/Match: {r['avg_crosses']:.1f}<br>"
                  f"Completion: {r['completion_rate']:.1f}%", axis=1),
    showlegend=False
), row=2, col=1)

# Update layout
fig.update_layout(
    title_text='FIFA World Cup 2022: Crossing Analysis Dashboard',
    height=900,
    plot_bgcolor='white',
    legend=dict(orientation="h", yanchor="bottom", y=1.02, xanchor="right", x=1)
)

# Update axes
fig.update_yaxes(title_text="Average Crosses", row=1, col=1)
fig.update_yaxes(title_text="Completion Rate (%)", row=2, col=1)
fig.update_xaxes(title_text="Tournament Stage", row=1, col=1)
fig.update_xaxes(title_text="Average Crosses per Match", row=2, col=1)
fig.update_yaxes(
    title_text="Percentage (%)",
    overlaying='y',
    side='right',
    row=1, col=1
)

fig.show()
```

1.4.14 Final Match

[71]:
```
import pandas as pd
import plotly.graph_objects as go
from plotly.subplots import make_subplots

final_match = df[df['category'] == 'Final'].iloc[0] if 'Final' in␣
 ↪df['category'].values else None

if final_match is None:
    print("Final match data not found")
    exit()
```

```
# Team names
team1, team2 = final_match['team1'], final_match['team2']

# Metric configuration
metrics = [
    ('Possession (%)', round(final_match['possession team1']*100, 1),␣
 ↪round(final_match['possession team2']*100, 1)),
    ('Goals', final_match['number of goals team1'], final_match['number of␣
 ↪goals team2']),
    ('Total Attempts', final_match['total attempts team1'], final_match['total␣
 ↪attempts team2']),
    ('On Target', final_match['on target attempts team1'], final_match['on␣
 ↪target attempts team2']),
    ('Off Target', final_match['off target attempts team1'], final_match['off␣
 ↪target attempts team2']),
    ('Crosses', final_match['crosses team1'], final_match['crosses team2']),
    ('Completed Crosses', final_match['crosses completed team1'],␣
 ↪final_match['crosses completed team2']),
    ('Cross Completion (%)', round(final_match['crosses completed team1']/
 ↪final_match['crosses team1']*100, 1) if final_match['crosses team1'] > 0␣
 ↪else 0,
     round(final_match['crosses completed team2']/final_match['crosses␣
 ↪team2']*100, 1) if final_match['crosses team2'] > 0 else 0),
    ('Passes', final_match['passes team1'], final_match['passes team2']),
    ('Pass Completion (%)', round(final_match['passes completed team1']/
 ↪final_match['passes team1']*100, 1),
     round(final_match['passes completed team2']/final_match['passes␣
 ↪team2']*100, 1)),
    ('Corners', final_match['corners team1'], final_match['corners team2']),
    ('Fouls', final_match['fouls against team2'], final_match['fouls against␣
 ↪team1']),
    ('Yellow Cards', final_match['yellow cards team1'], final_match['yellow␣
 ↪cards team2'])
]

# Create DataFrame
stats_df = pd.DataFrame({
    'Metric': [m[0] for m in metrics],
    team1: [m[1] for m in metrics],
    team2: [m[2] for m in metrics]
})

# Create figure
fig = make_subplots(
    rows=2, cols=2,
    specs=[[{"type": "table"}, {"type": "bar"}],
           [{"type": "bar", "colspan": 2}, None]],
    subplot_titles=("Match Statistics", "Attack Comparison", "Distribution␣
  ↪Performance")
)

# Add table
fig.add_trace(
    go.Table(
        header=dict(values=stats_df.columns, fill_color='royalblue',␣
  ↪font=dict(color='white')),
        cells=dict(values=[stats_df[col] for col in stats_df.columns],␣
  ↪fill_color='whitesmoke')
    ),
    row=1, col=1
)

# Add attack comparison bars
attack_metrics = metrics[1:5]
for i, (team, color) in enumerate([(team1, '#75AADB'), (team2, '#E63946')]):
    fig.add_trace(
        go.Bar(
            x=[m[0] for m in attack_metrics],
            y=[stats_df[team].iloc[i+1] for i in range(4)],
            name=team,
            marker_color=color
        ),
        row=1, col=2
    )

# Add distribution bars
dist_metrics = metrics[5:10]
for i, (team, color) in enumerate([(team1, '#75AADB'), (team2, '#E63946')]):
    fig.add_trace(
        go.Bar(
            x=[m[0] for m in dist_metrics],
            y=[stats_df[team].iloc[i+5] for i in range(5)],
            name=team,
            marker_color=color,
            showlegend=False
        ),
        row=2, col=1
    )

# Update layout
fig.update_layout(
    title=f"World Cup Final: {team1} vs {team2}",
    height=700,
    barmode='group',
    plot_bgcolor='white',
    legend=dict(orientation='h', y=1.1)
)

fig.show()
```
