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

# Load data
df = pd.read_excel("Women Lacrosse DataSet.xlsx")
df.columns = df.columns.str.strip().str.replace(" ", "_")

# Split 'GP-GS' into two columns
if 'GP-GS' in df.columns:
    df[['GP', 'GS']] = df['GP-GS'].str.split("-", expand=True)
    df.drop(columns=['GP-GS'], inplace=True)

# Split 'RC-YC-GC' into three columns
if 'RC-YC-GC' in df.columns:
    df[['RC', 'YC', 'GC']] = df['RC-YC-GC'].str.split("-",
expand=True)
    df.drop(columns=['RC-YC-GC'], inplace=True)

# Convert all numeric columns to numbers
for col in df.columns:
    if col != 'PLAYER':  # Keep PLAYER as string
        df[col] = pd.to_numeric(df[col], errors='coerce')

# Show cleaned data
print("Cleaned Data Sample:")
display(df.head())

# Summary statistics (numeric only)
print("Cleaned Summary Statistics:")
display(df.describe())
```

```
Cleaned Data Sample:
```

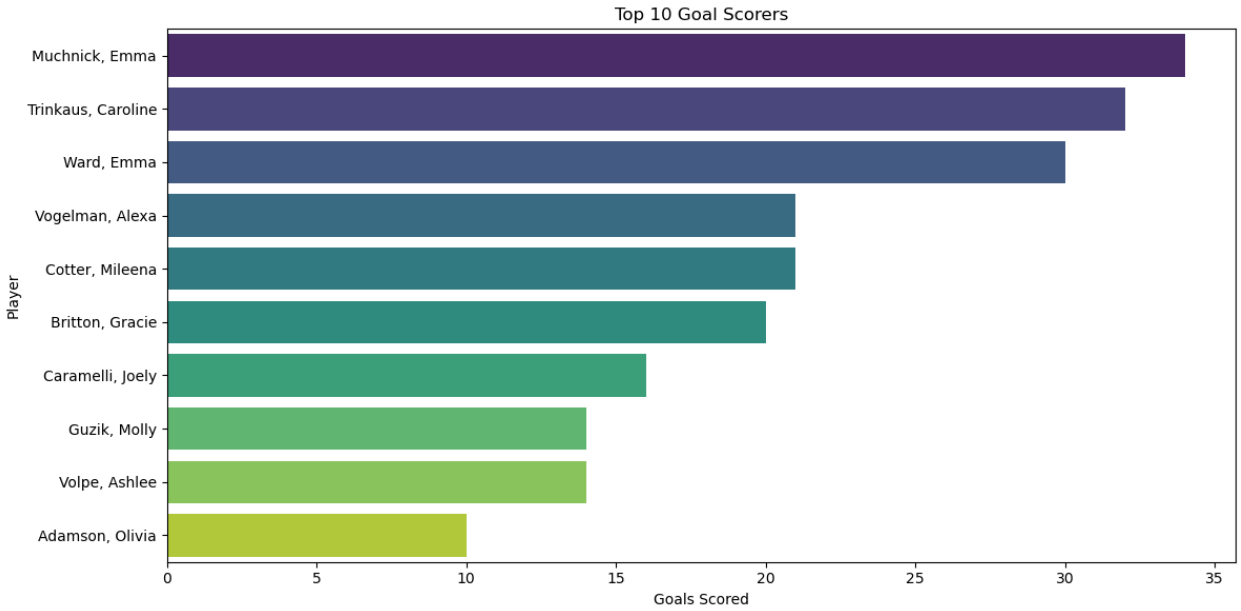
	NUMBER	PLAYER	G	A	PTS	SH	SH%	SOG	SOG%	GWG	...
0	44	Ward, Emma	30	46	76	77	0.390	55	0.714	1	...
1	24	Trinkaus, Caroline	32	11	43	72	0.444	57	0.792	4	...
2	5	Muchnick, Emma	34	7	41	71	0.479	55	0.775	2	...
3	19	Britton, Gracie	20	10	30	41	0.488	33	0.805	0	...
4	11	Vogelman, Alexa	21	6	27	46	0.457	35	0.761	0	...

	GB	TO	CT	DC	FOULS	GP	GS	RC	YC	GC

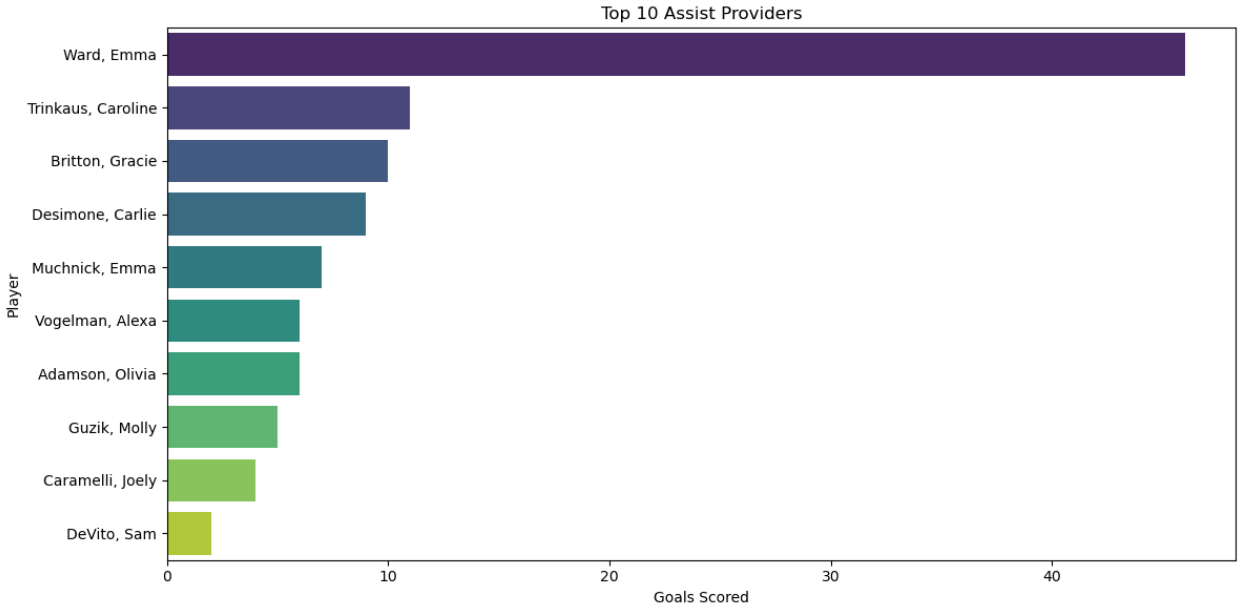
```
0   6  41   2   0      9  19.0  19.0   0   1   0
1   6  16   5   8      6  19.0  18.0   0   3   7
2  27  31   9  13      8  19.0  18.0   0   1   1
3   8  16   0   1      2  19.0  14.0   0   0   1
4  25  27  13  31     26  19.0  10.0   0   3   0

[5 rows x 22 columns]

Cleaned Summary Statistics:

          NUMBER          G          A        PTS         SH        SH
%  \
count  32.000000  32.000000  32.000000  32.000000  32.000000
32.000000
mean   23.281250   7.312500   3.500000  10.812500  16.812500
0.254938
std    19.804259  10.648148   8.428064  17.321323  24.400473
0.273217
min     0.000000   0.000000   0.000000   0.000000   0.000000
0.000000
25%     8.750000   0.000000   0.000000   0.000000   0.000000
0.000000
50%    19.500000   1.000000   0.000000   1.000000   2.500000
0.291500
75%    30.750000  14.000000   4.250000  16.750000  31.750000
0.453250
max    88.000000  34.000000  46.000000  76.000000  77.000000
1.000000

             SOG       SOG%        GWG        FPG  ...         GB
TO  \
count  32.000000  32.000000  32.000000  32.000000  ...  32.000000
32.000000
mean   12.531250   0.485563   0.312500   1.375000  ...   9.187500
7.968750
std    18.282698   0.397535   0.820602   3.034745  ...  10.855168
10.322179
min     0.000000   0.000000   0.000000   0.000000  ...   0.000000
0.000000
25%     0.000000   0.000000   0.000000   0.000000  ...   1.000000
0.750000
50%     2.000000   0.671500   0.000000   0.000000  ...   5.500000
4.000000
75%    22.250000   0.764500   0.000000   1.000000  ...  14.250000
10.250000
max    57.000000   1.000000   4.000000  12.000000  ...  34.000000
41.000000

              CT         DC      FOULS         GP         GS    RC
YC  \
```

```
count  32.000000  32.000000  32.000000  16.000000  16.000000  32.0
32.000000
mean    4.781250   7.468750  11.187500  18.187500  12.562500   0.0
0.718750
std     8.019067  15.682298  15.264205   1.515201   6.366252   0.0
0.888434
min     0.000000   0.000000   0.000000  15.000000   0.000000   0.0
0.000000
25%     0.000000   0.000000   0.000000  18.500000   9.750000   0.0
0.000000
50%     1.000000   0.500000   3.500000  19.000000  13.000000   0.0
0.500000
75%     8.250000   8.750000  15.250000  19.000000  18.000000   0.0
1.000000
max    40.000000  75.000000  51.000000  19.000000  19.000000   0.0
3.000000

               GC
count  32.000000
mean    1.250000
std     1.759765
min     0.000000
25%     0.000000
50%     1.000000
75%     1.250000
max     7.000000

[8 rows x 21 columns]
```

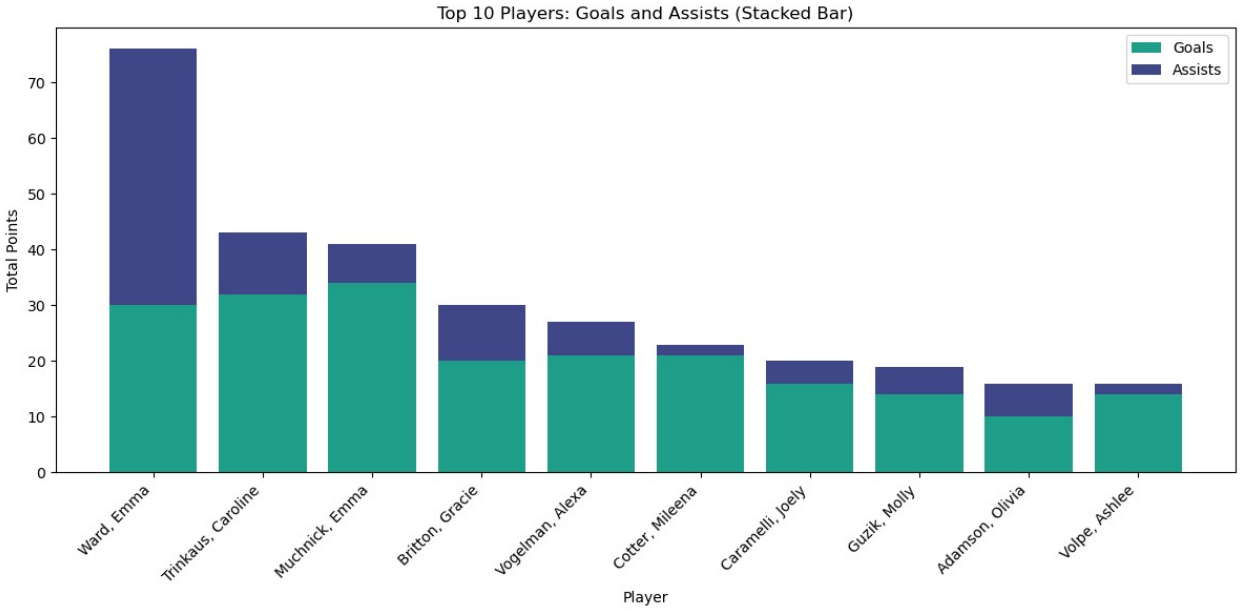
```python
# Top 10 Goal Scorers (Bar Plot)
plt.figure(figsize=(12, 6))
top_scorers = df.sort_values(by="G", ascending=False).head(10)
sns.barplot(data=top_scorers, x="G", y="PLAYER", palette="viridis")
plt.title("Top 10 Goal Scorers")
plt.xlabel("Goals Scored")
plt.ylabel("Player")
plt.tight_layout()
plt.show()
```




```
# Top 10 Assist Providers (Bar Plot)
plt.figure(figsize=(12, 6))
top_scorers = df.sort_values(by="A", ascending=False).head(10)
sns.barplot(data=top_scorers, x="A", y="PLAYER", palette="viridis")
plt.title("Top 10 Assist Providers")
plt.xlabel("Goals Scored")
plt.ylabel("Player")
plt.tight_layout()
plt.show()
```

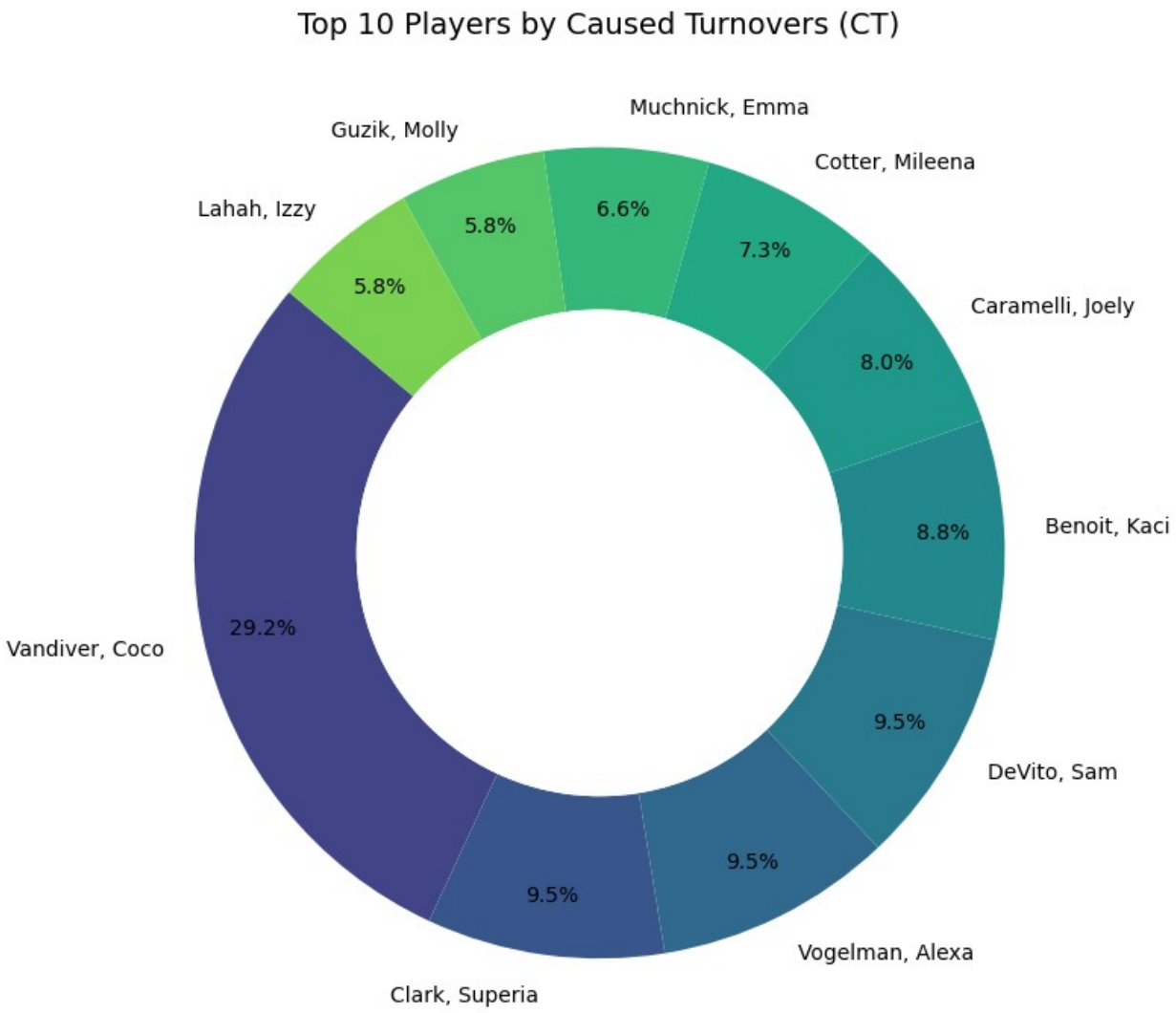


```
# Stacked Bar Plot with Custom Viridis Colors
plt.figure(figsize=(12, 6))
plt.bar(top10_points['PLAYER'], top10_points['G'], label='Goals',
color='#1f9e89')      # Dark green
plt.bar(top10_points['PLAYER'], top10_points['A'],
bottom=top10_points['G'], label='Assists', color='#3f4788')  # Deep
blue-violet
plt.xticks(rotation=45, ha='right')
plt.xlabel("Player")
plt.ylabel("Total Points")
plt.title("Top 10 Players: Goals and Assists (Stacked Bar)")
plt.legend()
plt.tight_layout()
plt.show()
```




```
# Donut Chart for CT (Caused Turnovers)
top_ct = df[df['CT'] > 0].sort_values(by='CT',
ascending=False).head(10)

plt.figure(figsize=(8, 8))
colors = plt.cm.viridis(np.linspace(0.2, 0.8, len(top_ct)))

plt.pie(
    top_ct['CT'],
    labels=top_ct['PLAYER'],
    colors=colors,
    startangle=140,
    wedgeprops=dict(width=0.4),  # Donut effect
    autopct='%1.1f%%',
    pctdistance=0.85
```

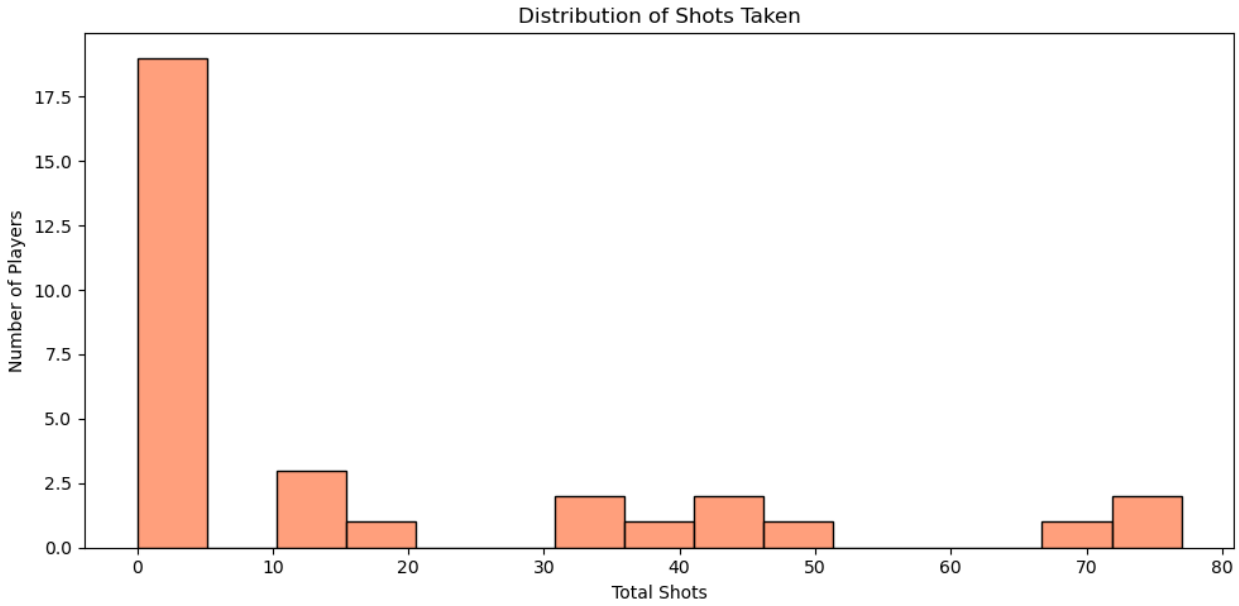
```
)

# Add white circle in center for donut look
centre_circle = plt.Circle((0, 0), 0.6, fc='white')
plt.gca().add_artist(centre_circle)

plt.title("Top 10 Players by Caused Turnovers (CT)", fontsize=14)
plt.tight_layout()
plt.show()
```




```
# Histogram: Distribution of Shots Taken
plt.figure(figsize=(10, 5))
sns.histplot(df["SH"], bins=15, color='coral')
plt.title("Distribution of Shots Taken")
```

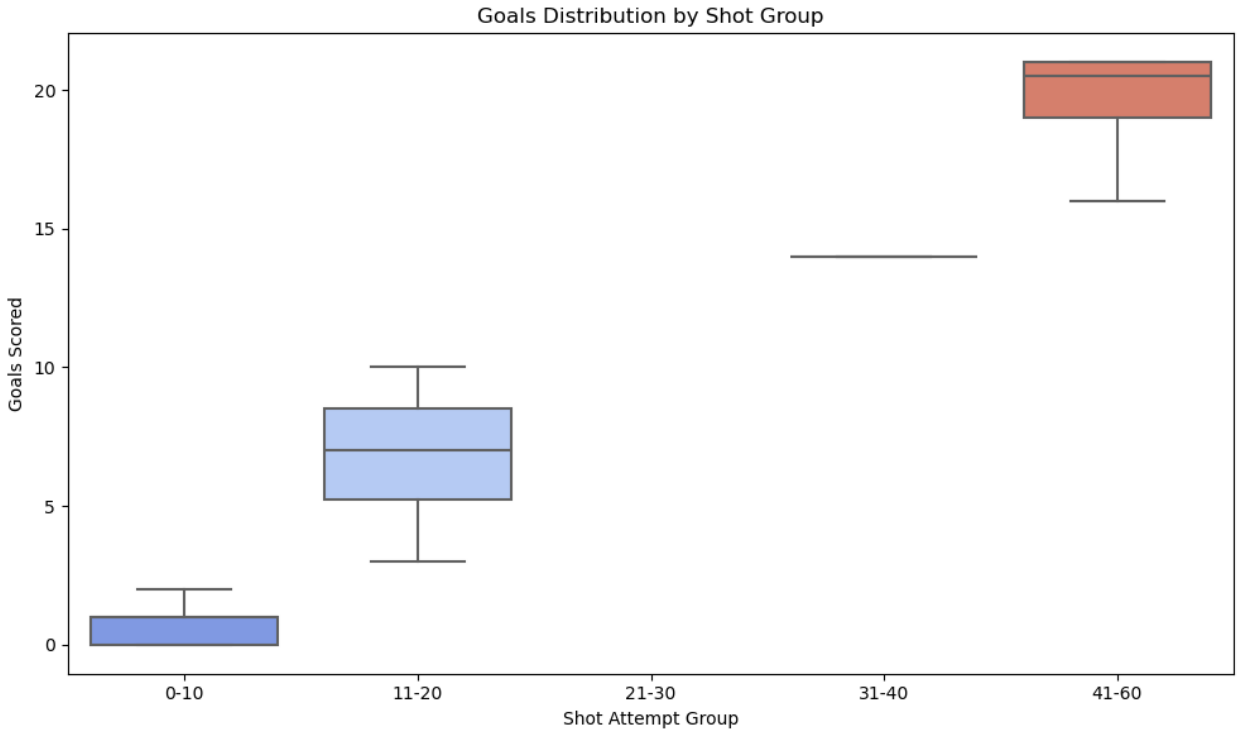
```
plt.xlabel("Total Shots")
plt.ylabel("Number of Players")
plt.tight_layout()
plt.show()
```

```
C:\Users\kunal\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
# Box Plot: Goals vs Shot Attempts
df['Shot_Group'] = pd.cut(df['SH'], bins=[0, 10, 20, 30, 40, 60],
labels=["0-10", "11-20", "21-30", "31-40", "41-60"])
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x="Shot_Group", y="G", palette="coolwarm")
plt.title("Goals Distribution by Shot Group")
plt.xlabel("Shot Attempt Group")
plt.ylabel("Goals Scored")
plt.tight_layout()
plt.show()
```

```
C:\Users\kunal\anaconda3\Lib\site-packages\seaborn\categorical.py:641:
FutureWarning: The default of observed=False is deprecated and will be
changed to True in a future version of pandas. Pass observed=False to
retain current behavior or observed=True to adopt the future default
and silence this warning.
  grouped_vals = vals.groupby(grouper)
```


Goals Distribution by Shot Group
Goals Scored
0
5
10
15
20
0-10
11-20
21-30
31-40
41-60
Shot Attempt Group