P1

November 13, 2024

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

0.1 Basic EDA on Coco Gauff

[2]:
```
summary = pd.read_csv('CocoG.csv')

summary
```

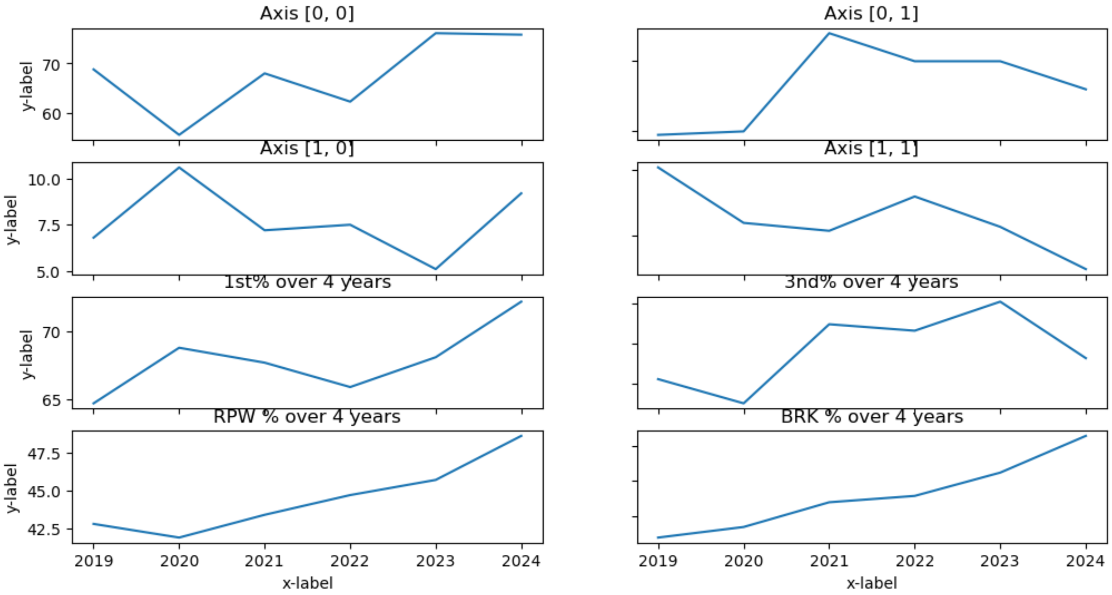
[2]:
```
   Year   M   W   L   Win% Set W-L   Set% Game W-L  Game% TB W-L  ...    A%  \
0  2024  66  50  16  75.8%  106-44  70.7%  808-539  60.0%    8-7  ...  5.2%
1  2023  67  51  16  76.1%  107-43  71.3%  776-563  58.0%    8-1  ...  6.0%
2  2022  61  38  23  62.3%   80-51  61.1%  672-572  54.0%   10-7  ...  6.0%
3  2021  50  34  16  68.0%   69-46  60.0%  590-514  53.4%   10-7  ...  6.8%
4  2020  18  10   8  55.6%   26-19  57.8%  229-217  51.3%    3-2  ...  4.0%
5  2019  16  11   5  68.8%   21-16  56.8%  172-176  49.4%    3-0  ...  3.9%

     DF%  1stIn   1st%   2nd%    SPW    RPW    TPW    DR               Best
0   9.2%  57.5%  72.2%  44.1%  60.3%  48.6%  54.4%  1.22             W (2x)
1   5.1%  60.7%  68.1%  47.6%  60.1%  45.7%  52.9%  1.14             W (4x)
2   7.5%  63.0%  65.9%  45.8%  58.5%  44.7%  51.5%  1.08  F (Roland Garros)
3   7.2%  60.4%  67.7%  46.2%  59.2%  43.4%  51.3%  1.06          W (Parma)
4  10.6%  61.0%  68.8%  41.3%  58.1%  41.9%  50.2%  1.00     SF (Lexington)
5   6.8%  65.2%  64.7%  42.8%  57.1%  42.8%  49.9%  1.00           W (Linz)

[6 rows x 24 columns]
```

[3]:
```
summary.columns
```

[3]:
```
Index(['Year', 'M', 'W', 'L', 'Win%', 'Set W-L', 'Set%', 'Game W-L', 'Game%',
       'TB W-L', 'TB%', 'MS', 'Hld%', 'Brk%', 'A%', 'DF%', '1stIn', '1st%',
       '2nd%', 'SPW', 'RPW', 'TPW', 'DR', 'Best'],
      dtype='object')
```

[4]:
```
fig, axs = plt.subplots(4, 2, figsize=(12, 6))
axs[0, 0].plot(summary['Year'], summary['Win'])
```

```
axs[0, 0].set_title('Axis [0, 0]')
axs[0, 1].plot(summary['Year'], summary['A'])
axs[0, 1].set_title('Axis [0, 1]')
axs[1, 0].plot(summary['Year'], summary['DF'])
axs[1, 0].set_title('Axis [1, 0]')
axs[1, 1].plot(summary['Year'], summary['1stIn'])
axs[1, 1].set_title('Axis [1, 1]')

axs[2, 0].plot(summary['Year'], summary['1st'])
axs[2, 0].set_title('1st% over 4 years')

axs[2, 1].plot(summary['Year'], summary['2nd'])
axs[2, 1].set_title('3nd% over 4 years')

axs[3, 0].plot(summary['Year'], summary['RPW'])
axs[3, 0].set_title('RPW % over 4 years')

axs[3, 1].plot(summary['Year'], summary['Brk'])
axs[3, 1].set_title('BRK % over 4 years')



for ax in axs.flat:
    ax.set(xlabel='x-label', ylabel='y-label')

# Hide x labels and tick labels for top plots and y ticks for right plots.
for ax in axs.flat:
    ax.label_outer()
```



0.2 More detailed Data

[7]:

```
y2019 = pd.read_csv('wta_matches_2019.csv')
y2020 = pd.read_csv('wta_matches_2020.csv')
y2021 = pd.read_csv('wta_matches_2021.csv')
y2022 = pd.read_csv('wta_matches_2022.csv')
y2023 = pd.read_csv('wta_matches_2023.csv')
y2024 = pd.read_csv('wta_matches_2024.csv')
```

[182]:

```
Cgwin2019 = y2019[y2019['winner_name'] == 'Coco Gauff']
Cglose2019 = y2019[y2019['loser_name'] == 'Coco Gauff']

Cgwin2020 = y2020[y2020['winner_name'] == 'Coco Gauff']
Cglose2020 = y2020[y2020['loser_name'] == 'Coco Gauff']

Cgwin2021 = y2021[y2021['winner_name'] == 'Coco Gauff']
Cglose2021 = y2021[y2021['loser_name'] == 'Coco Gauff']

Cgwin2022 = y2022[y2022['winner_name'] == 'Coco Gauff']
Cglose2022 = y2022[y2022['loser_name'] == 'Coco Gauff']

Cgwin2023 = y2023[y2023['winner_name'] == 'Coco Gauff']
Cglose2023 = y2023[y2023['loser_name'] == 'Coco Gauff']

Cgwin2024 = y2024[y2024['winner_name'] == 'Coco Gauff']
Cglose2024 = y2024[y2024['loser_name'] == 'Coco Gauff']
```

[183]:

```
Cgyears = pd.concat([Cgwin2019, Cglose2019,Cgwin2020, Cglose2020, Cgwin2021,␣
  ↪Cglose2021, Cgwin2022,
                   Cglose2022, Cgwin2023, Cglose2023, Cgwin2024, Cglose2024],␣
  ↪axis=0)

Cgyears['Year'] = Cgyears['tourney_date'].astype(str).str[:4]

Cgyears['Win'] = Cgyears['winner_name'].apply(lambda x: 0 if x != 'Coco Gauff'␣
  ↪else 1)
Cgyears['Lose'] = Cgyears['loser_name'].apply(lambda x: 0 if x != 'Coco Gauff'␣
  ↪else 1)
Cgyears['Outcome'] = Cgyears['loser_name'].apply(lambda x: 'Lose' if x != 'Coco␣
  ↪Gauff' else 'Win')

# Calculate Win Percentage
Cgyears['Win_Percentage'] = Cgyears['Win'] / (Cgyears['Win'] + Cgyears['Lose'])
```

0.3 Comparing against Top player

[121]:

```
aswin2019 = y2019[y2019['winner_name'] == 'Aryna Sabalenka']
aslose2019 = y2019[y2019['loser_name'] == 'Aryna Sabalenka']

aswin2020 = y2020[y2020['winner_name'] == 'Aryna Sabalenka']
aslose2020 = y2020[y2020['loser_name'] == 'Aryna Sabalenka']

aswin2021 = y2021[y2021['winner_name'] == 'Aryna Sabalenka']
aslose2021 = y2021[y2021['loser_name'] == 'Aryna Sabalenka']

aswin2022 = y2022[y2022['winner_name'] == 'Aryna Sabalenka']
aslose2022 = y2022[y2022['loser_name'] == 'Aryna Sabalenka']

aswin2023 = y2023[y2023['winner_name'] == 'Aryna Sabalenka']
aslose2023 = y2023[y2023['loser_name'] == 'Aryna Sabalenka']

aswin2024 = y2024[y2024['winner_name'] == 'Aryna Sabalenka']
aslose2024 = y2024[y2024['loser_name'] == 'Aryna Sabalenka']
```

[177]:

```
Asyears = pd.concat([aswin2019, aslose2019,aswin2020, aslose2020,aswin2021,␣
 ↪aslose2021, aswin2022,
                   aslose2022, aswin2023, aslose2023, aswin2024, aslose2024],␣
 ↪axis=0)

Asyears['Year'] = Asyears['tourney_date'].astype(str).str[:4]

Asyears['Win'] = Asyears['winner_name'].apply(lambda x: 0 if x != 'Aryna␣
 ↪Sabalenka' else 1)
Asyears['Lose'] = Asyears['loser_name'].apply(lambda x: 0 if x != 'Aryna␣
 ↪Sabalenka' else 1)


Asyears['Win_Percentage'] = Asyears['Win'] / (Asyears['Win'] + Asyears['Lose'])

Asyears = Asyears[Asyears['Year'] > '2018']
```

0.4 Comparison against similar standing Player #2 player

[133]:

```
qzwin2019 = y2019[y2019['winner_name'] == 'Qinwen Zheng']
qzlose2019 = y2019[y2019['loser_name'] == 'Qinwen Zheng']

qzwin2020 = y2020[y2020['winner_name'] == 'Qinwen Zheng']
qzlose2020 = y2020[y2020['loser_name'] == 'Qinwen Zheng']

qzwin2021 = y2021[y2021['winner_name'] == 'Qinwen Zheng']
qzlose2021 = y2021[y2021['loser_name'] == 'Qinwen Zheng']
```

```
qzwin2022 = y2022[y2022['winner_name'] == 'Qinwen Zheng']
qzlose2022 = y2022[y2022['loser_name'] == 'Qinwen Zheng']

qzwin2023 = y2023[y2023['winner_name'] == 'Qinwen Zheng']
qzlose2023 = y2023[y2023['loser_name'] == 'Qinwen Zheng']

qzwin2024 = y2024[y2024['winner_name'] == 'Qinwen Zheng']
qzlose2024 = y2024[y2024['loser_name'] == 'Qinwen Zheng']
```

[134]:
```
qzyears = pd.concat([qzwin2019, qzlose2019,qzwin2020, qzlose2020,qzwin2021,␣
 ↪qzlose2021, qzwin2022,
                   qzlose2022, qzwin2023, qzlose2023, qzwin2024, qzlose2024],␣
 ↪axis=0)

qzyears['Year'] = qzyears['tourney_date'].astype(str).str[:4]

qzyears['Win'] = qzyears['winner_name'].apply(lambda x: 0 if x != 'Qinwen␣
 ↪Zheng' else 1)
qzyears['Lose'] = qzyears['loser_name'].apply(lambda x: 0 if x != 'Qinwen␣
 ↪Zheng' else 1)


qzyears['Win_Percentage'] = qzyears['Win'] / (qzyears['Win'] + qzyears['Lose'])
```

0.5 Against Surface Overall Win Percentage

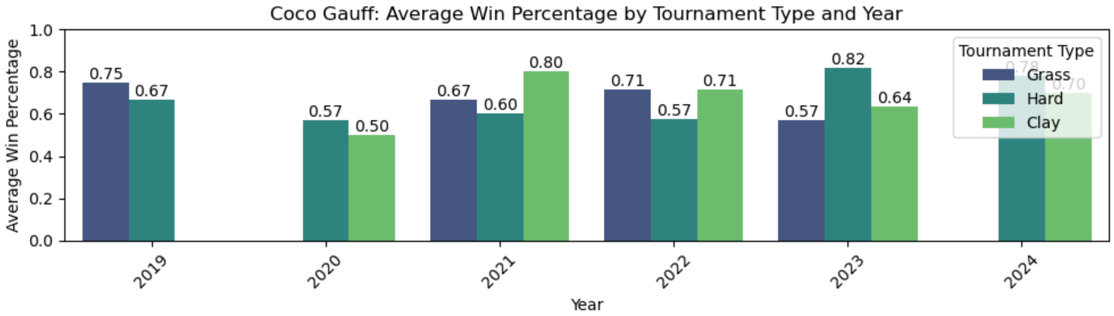
[184]:
```
# Group by Year and Surface, and calculate average win percentage

Cgperformance_summary = Cgyears.groupby(['Year', 'surface']).
 ↪agg({'Win_Percentage': 'mean'}).reset_index()

Asperformance_summary = Asyears.groupby(['Year', 'surface']).
 ↪agg({'Win_Percentage': 'mean'}).reset_index()

qzperformance_summary = qzyears.groupby(['Year', 'surface']).
 ↪agg({'Win_Percentage': 'mean'}).reset_index()
```

[185]:
```
Cgperformance_summary['Player'] = 'Coco Gauff'
Asperformance_summary['Player'] = 'Aryna Sabalenka'
qzperformance_summary['Player'] = 'Qinwen Zheng'
```

[186]:
```
combined_summary = pd.concat([Cgperformance_summary, Asperformance_summary,␣
 ↪qzperformance_summary])
```

[162]:

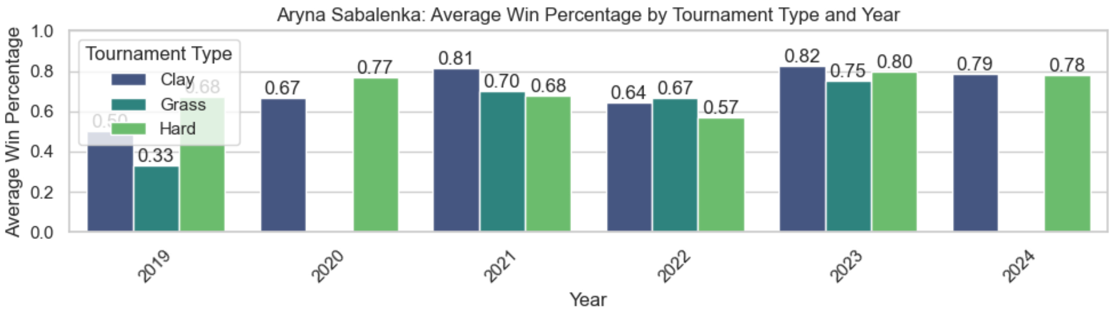
```
# Calculate average Win_Percentage by Year and Type
Cgaverage_win_percentage = Cgperformance_summary.groupby(['Year',␣
 ↪'surface'])['Win_Percentage'].mean().reset_index()

# Plotting
plt.figure(figsize=(10, 3))
sns.barplot(data=Cgaverage_win_percentage, x='Year', y='Win_Percentage',␣
 ↪hue='surface', palette='viridis')

# Customize the plot
plt.title('Coco Gauff: Average Win Percentage by Tournament Type and Year')
plt.xlabel('Year')
plt.ylabel('Average Win Percentage')
plt.legend(title='Tournament Type')
plt.ylim(0, 1)  # Set y-axis limits from 0 to 1
plt.xticks(rotation=45)
plt.tight_layout()

# Adding data labels
for p in plt.gca().patches:
    plt.annotate(f'{p.get_height():.2f}',
                 (p.get_x() + p.get_width() / 2., p.get_height()),
                 ha='center', va='bottom')

plt.tight_layout()

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



[191]:

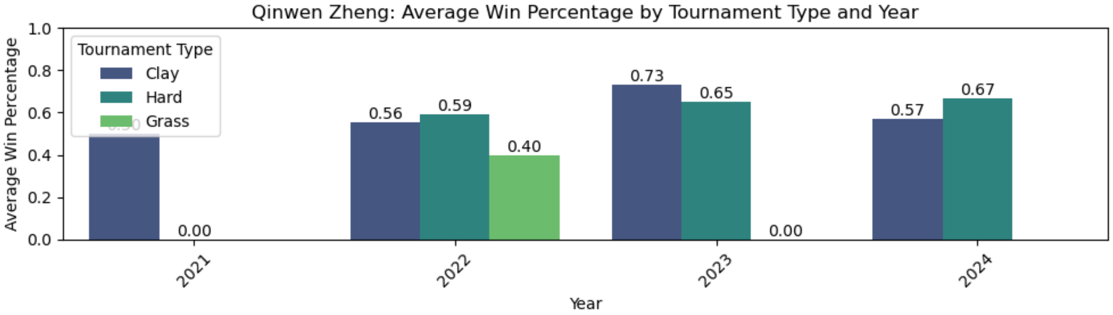
```
# Calculate average Win_Percentage by Year and Type
Asaverage_win_percentage = Asperformance_summary.groupby(['Year',␣
 ↪'surface'])['Win_Percentage'].mean().reset_index()

# Plotting
```

```
plt.figure(figsize=(10, 3))
sns.barplot(data=Asaverage_win_percentage, x='Year', y='Win_Percentage',␣
 ↪hue='surface', palette='viridis')

# Customize the plot
plt.title('Aryna Sabalenka: Average Win Percentage by Tournament Type and Year')
plt.xlabel('Year')
plt.ylabel('Average Win Percentage')
plt.legend(title='Tournament Type')
plt.ylim(0, 1)  # Set y-axis limits from 0 to 1
plt.xticks(rotation=45)
plt.tight_layout()

# Adding data labels
for p in plt.gca().patches:
    plt.annotate(f'{p.get_height():.2f}',
                 (p.get_x() + p.get_width() / 2., p.get_height()),
                 ha='center', va='bottom')

plt.tight_layout()

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

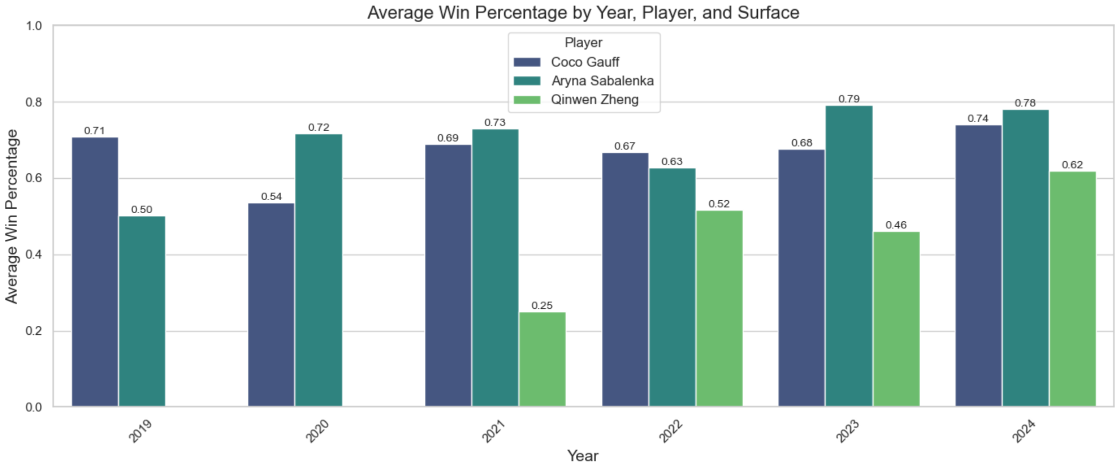


[160]:
```
# Calculate average Win_Percentage by Year and Type
qzaverage_win_percentage = qzperformance_summary.groupby(['Year',␣
 ↪'surface'])['Win_Percentage'].mean().reset_index()

# Plotting
plt.figure(figsize=(10, 3))
sns.barplot(data=qzaverage_win_percentage, x='Year', y='Win_Percentage',␣
 ↪hue='surface', palette='viridis')

# Customize the plot
```

```
plt.title('Qinwen Zheng: Average Win Percentage by Tournament Type and Year')
plt.xlabel('Year')
plt.ylabel('Average Win Percentage')
plt.legend(title='Tournament Type')
plt.ylim(0, 1)  # Set y-axis limits from 0 to 1
plt.xticks(rotation=45)
plt.tight_layout()

# Adding data labels
for p in plt.gca().patches:
    plt.annotate(f'{p.get_height():.2f}',
                 (p.get_x() + p.get_width() / 2., p.get_height()),
                 ha='center', va='bottom')

plt.tight_layout()

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



[196]:
```
# Set the style for the plot
sns.set(style="whitegrid")

# Plotting
plt.figure(figsize=(14, 6))
sns.barplot(data=combined_summary, x='Year', y='Win_Percentage', hue='Player',␣
 ↪palette='viridis', errorbar=None)

# Customize the plot
plt.title('Average Win Percentage by Year, Player, and Surface', fontsize=16)
plt.xlabel('Year', fontsize=14)
plt.ylabel('Average Win Percentage', fontsize=14)
plt.legend(title='Player', fontsize=12)
plt.ylim(0, 1)  # Set y-axis limits from 0 to 1
plt.xticks(rotation=45)
plt.tight_layout()

# Adding data labels on top of the bars
for p in plt.gca().patches:
    plt.annotate(f'{p.get_height():.2f}',
                 (p.get_x() + p.get_width() / 2., p.get_height()),
                 ha='center', va='bottom', fontsize=10)

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



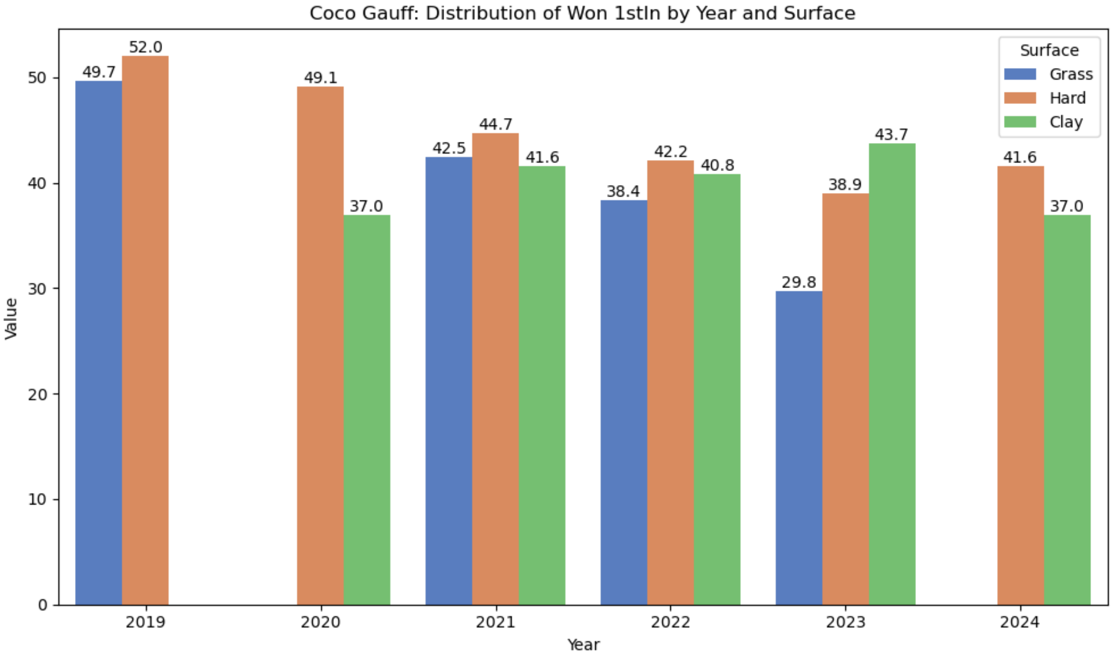
0.6 Against Surface Serve %

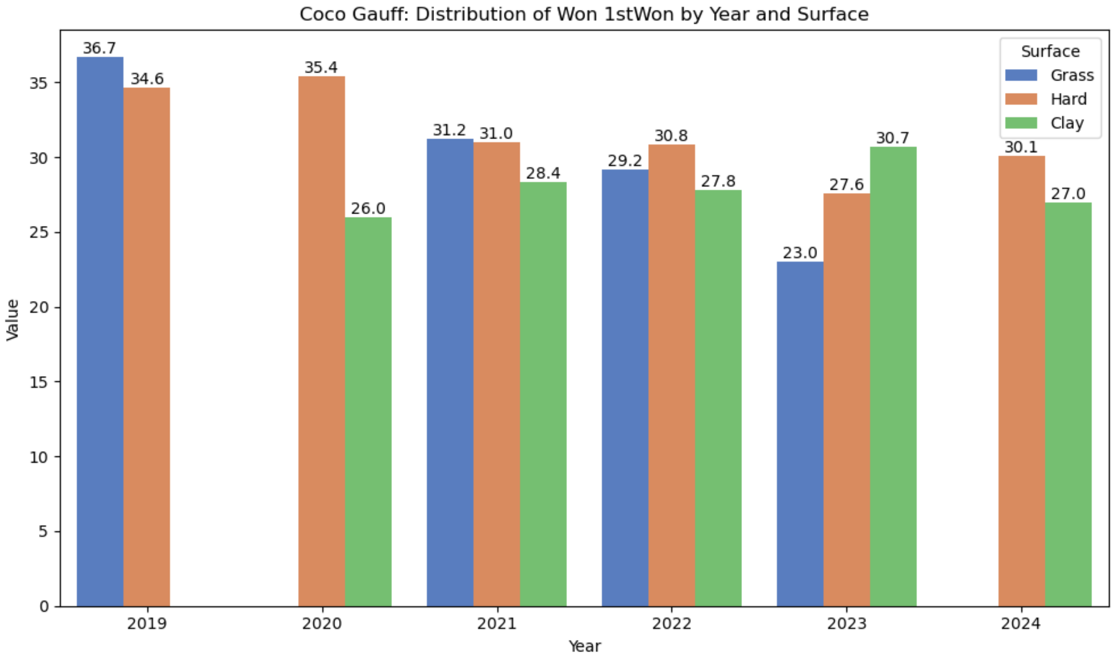
[139]:
```
## look into w_1stIn        w_1stWon        w_2ndWon

CGw = Cgyears[Cgyears['Win'] == 1]
CGl = Cgyears[Cgyears['Lose'] == 1]

# Group by Year and Surface, and calculate average win percentage

CGw_summary = CGw.groupby(['Year', 'surface']).agg({'w_1stIn': 'mean',␣
 ↪'w_1stWon': 'mean', 'w_2ndWon': 'mean', 'w_ace': 'mean', 'w_df': 'mean'}).
 ↪reset_index()
CGw_summary.rename({'w_1stIn': '1stIn', 'w_1stWon': '1stWon', 'w_2ndWon':␣
 ↪'2ndWon', 'w_ace': 'ace', 'w_df': 'df'}, axis=1, inplace=True)
CGl_summary = CGl.groupby(['Year', 'surface']).agg({'l_1stIn': 'mean',␣
 ↪'l_1stWon': 'mean', 'l_2ndWon': 'mean', 'l_ace': 'mean', 'l_df': 'mean'}).
 ↪reset_index()
CGl_summary.rename({'l_1stIn': '1stIn', 'l_1stWon': '1stWon', 'l_2ndWon':␣
 ↪'2ndWon', 'l_ace': 'ace', 'l_df': 'df'}, axis=1, inplace=True)
```

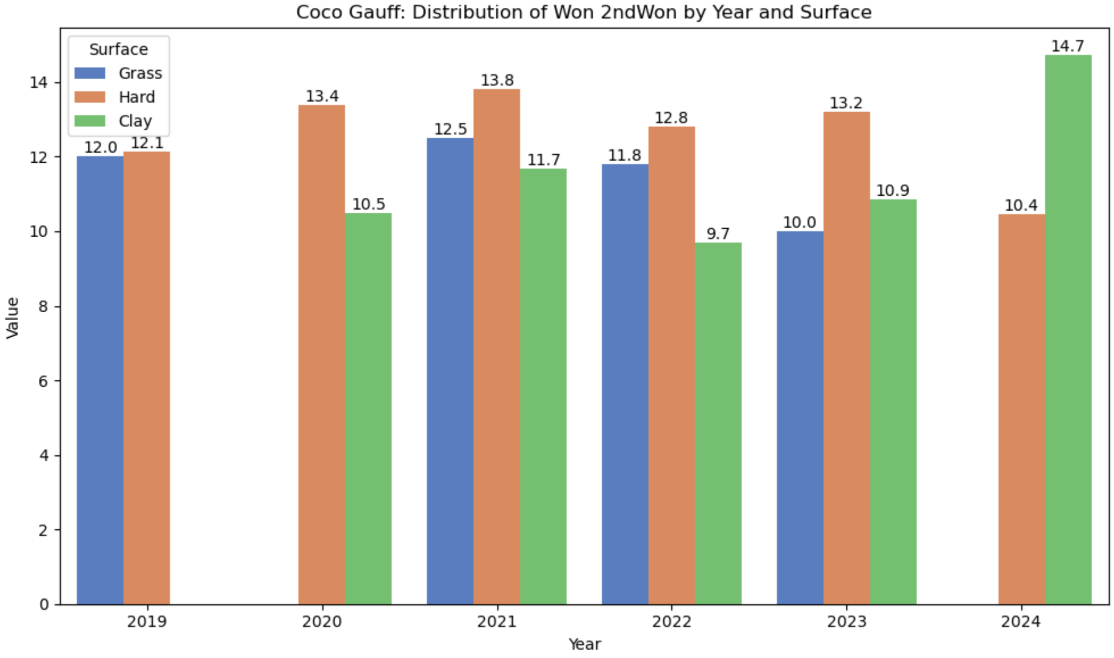
```
ASw = Asyears[Asyears['Win'] == 1]
ASl = Asyears[Asyears['Lose'] == 1]

# Group by Year and Surface, and calculate average win percentage
ASw_summary = ASw.groupby(['Year', 'surface']).agg({'w_1stIn': 'mean',␣
 ↪'w_1stWon': 'mean', 'w_2ndWon': 'mean', 'w_ace': 'mean', 'w_df': 'mean'}).
 ↪reset_index()
ASw_summary.rename({'w_1stIn': '1stIn', 'w_1stWon': '1stWon', 'w_2ndWon':␣
 ↪'2ndWon', 'w_ace': 'ace', 'w_df': 'df'}, axis=1, inplace=True)
ASl_summary = ASl.groupby(['Year', 'surface']).agg({'l_1stIn': 'mean',␣
 ↪'l_1stWon': 'mean', 'l_2ndWon': 'mean', 'l_ace': 'mean', 'l_df': 'mean'}).
 ↪reset_index()
ASl_summary.rename({'l_1stIn': '1stIn', 'l_1stWon': '1stWon', 'l_2ndWon':␣
 ↪'2ndWon', 'l_ace': 'ace', 'l_df': 'df'}, axis=1, inplace=True)

QZw = qzyears[qzyears['Win'] == 1]
QZl = qzyears[qzyears['Lose'] == 1]

# Group by Year and Surface, and calculate average win percentage
QZw_summary = QZw.groupby(['Year', 'surface']).agg({'w_1stIn': 'mean',␣
 ↪'w_1stWon': 'mean', 'w_2ndWon': 'mean', 'w_ace': 'mean', 'w_df': 'mean'}).
 ↪reset_index()
QZw_summary.rename({'w_1stIn': '1stIn', 'w_1stWon': '1stWon', 'w_2ndWon':␣
 ↪'2ndWon', 'w_ace': 'ace', 'w_df': 'df'}, axis=1, inplace=True)
QZl_summary = QZl.groupby(['Year', 'surface']).agg({'l_1stIn': 'mean',␣
 ↪'l_1stWon': 'mean', 'l_2ndWon': 'mean', 'l_ace': 'mean', 'l_df': 'mean'}).
 ↪reset_index()
QZl_summary.rename({'l_1stIn': '1stIn', 'l_1stWon': '1stWon', 'l_2ndWon':␣
 ↪'2ndWon', 'l_ace': 'ace', 'l_df': 'df'}, axis=1, inplace=True)
```

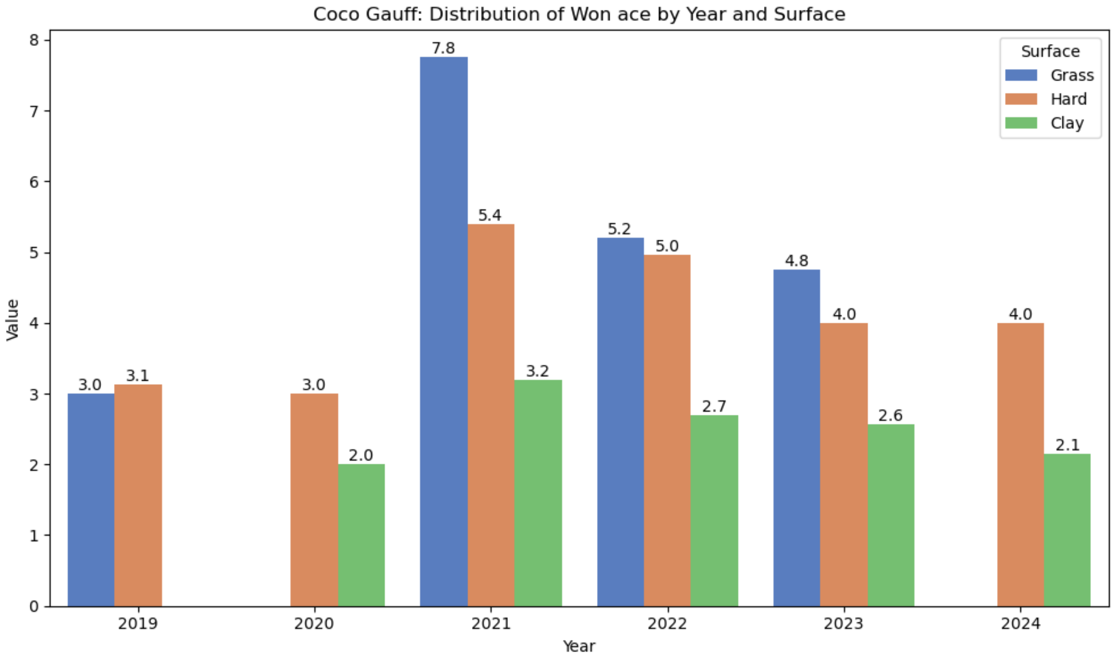
[140]:

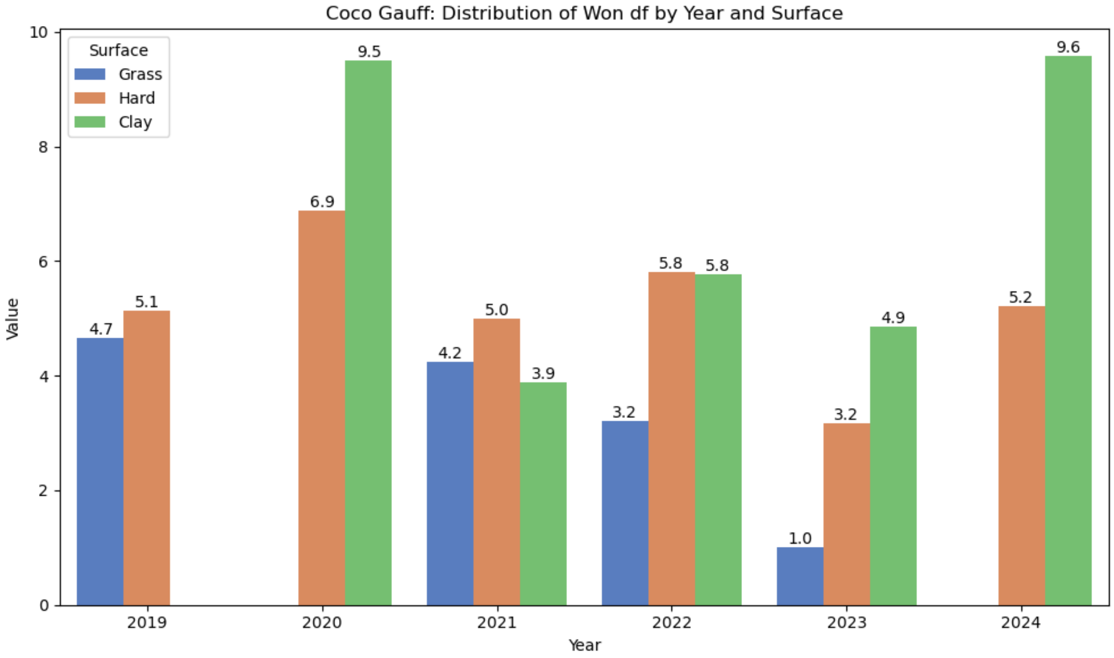
```
# Melting the DataFrame for easier plotting
CGw_melted = CGw_summary.melt(id_vars=['Year', 'surface'], value_vars=['1stIn',␣
 ↪'1stWon', '2ndWon', 'ace', 'df'],
                    var_name='Metric', value_name='Value')

metrics = ['1stIn', '1stWon', '2ndWon', 'ace', 'df']

for metric in metrics:
    plt.figure(figsize=(10, 6))
    ax = sns.barplot(data=CGw_melted[CGw_melted['Metric'] == metric], x='Year',␣
 ↪y='Value', hue='surface', errorbar=None, palette='muted')
    ax.set_title(f'Coco Gauff: Distribution of Won {metric} by Year and␣
 ↪Surface')
    ax.set_xlabel('Year')
    ax.set_ylabel('Value')
    ax.legend(title='Surface')

    # Adding data labels
    for p in ax.patches:
        ax.annotate(f'{p.get_height():.1f}',
                    (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom')

    plt.tight_layout()
    plt.show()
```



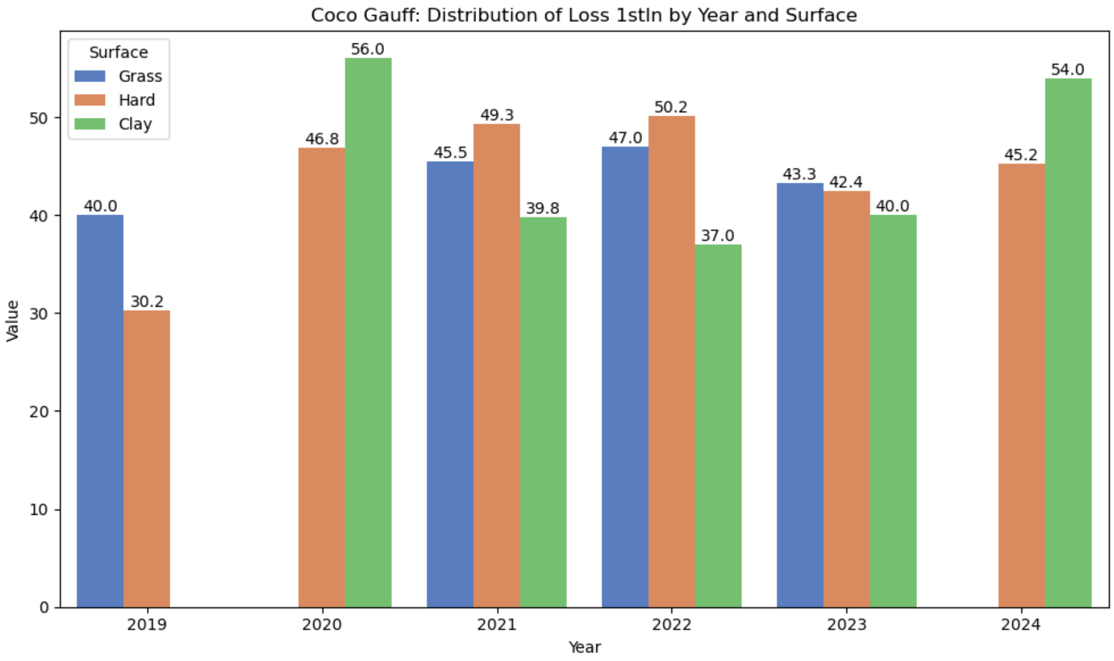

Coco Gauff: Distribution of Won 1stWon by Year and Surface
Surface
Grass
Hard
Clay
Value
Year
0
5
10
15
20
25
30
35
2019
2020
2021
2022
2023
2024
36.7
34.6
35.4
26.0
31.2
31.0
28.4
29.2
30.8
27.8
23.0
27.6
30.7
30.1
27.0

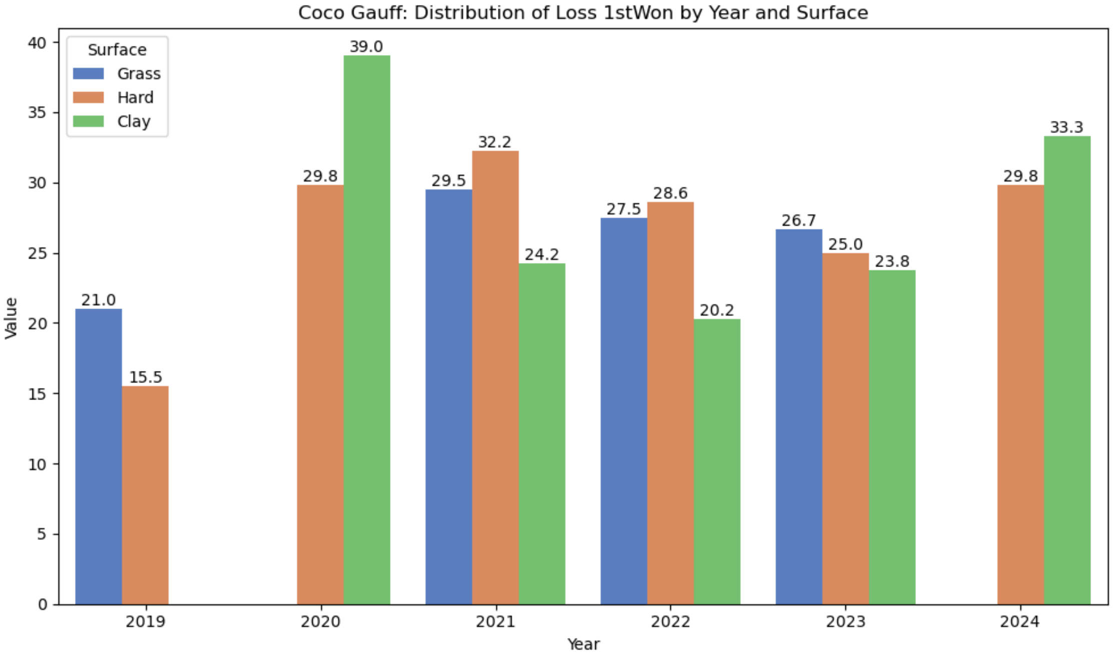

Coco Gauff: Distribution of Won 2ndWon by Year and Surface
Surface
Grass
Hard
Clay
Value
Year
0
2
4
6
8
10
12
14
2019
2020
2021
2022
2023
2024
12.0
12.1
13.4
10.5
12.5
13.8
11.7
11.8
12.8
9.7
10.0
13.2
10.9
10.4
14.7

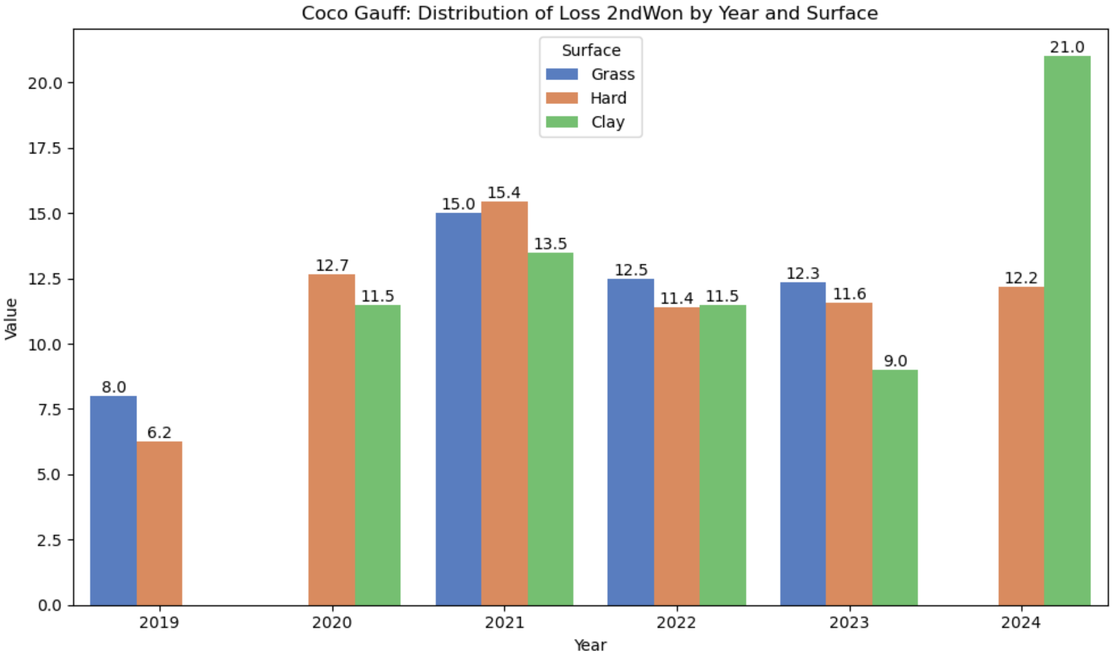


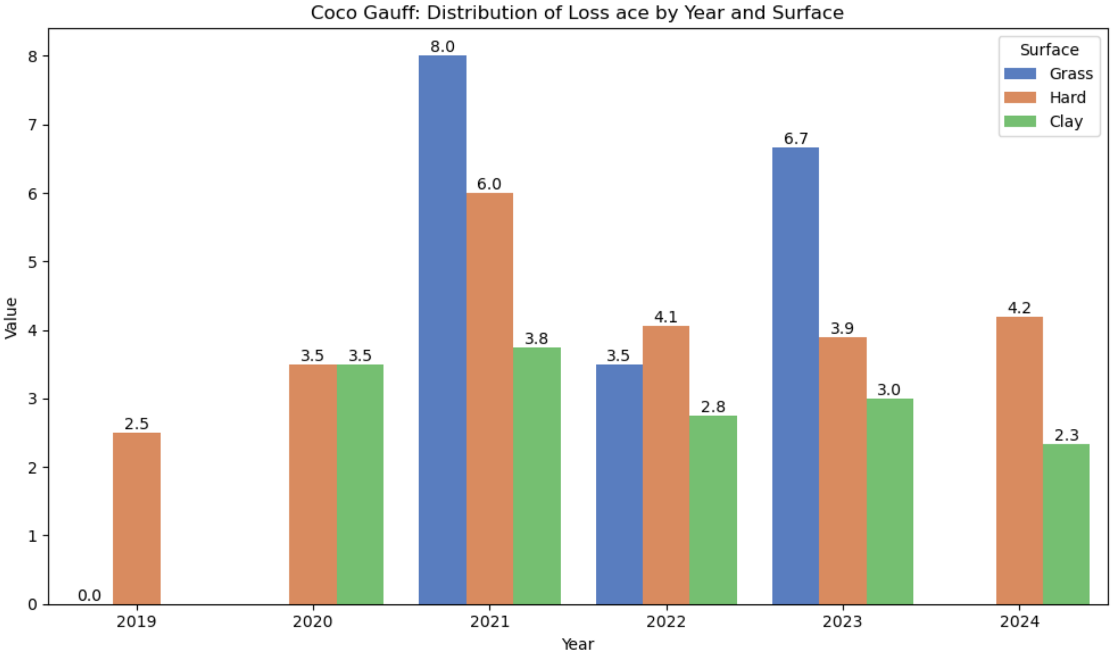


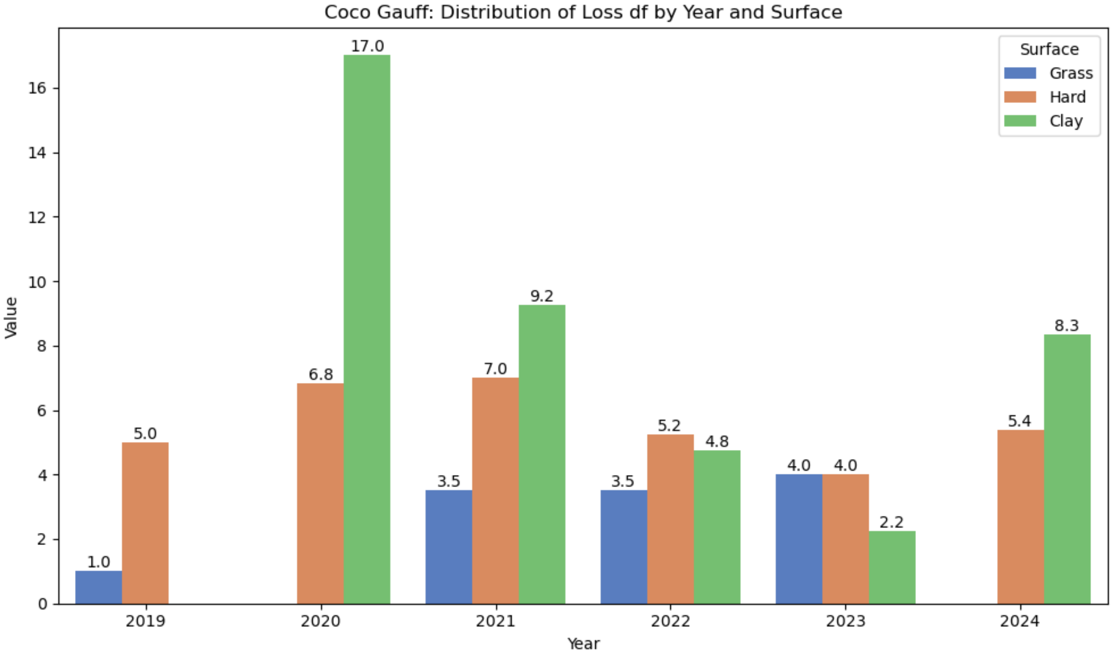
[141]:
```
# Melting the DataFrame for easier plotting
CGl_melted = CGl_summary.melt(id_vars=['Year', 'surface'], value_vars=['1stIn',␣
↪'1stWon', '2ndWon', 'ace', 'df'],
                         var_name='Metric', value_name='Value')

for metric in metrics:
    plt.figure(figsize=(10, 6))
    ax = sns.barplot(data=CGl_melted[CGl_melted['Metric'] == metric], x='Year',␣
 ↪y='Value', hue='surface', errorbar=None, palette='muted')
    ax.set_title(f'Coco Gauff: Distribution of Loss {metric} by Year and␣
 ↪Surface')
    ax.set_xlabel('Year')
    ax.set_ylabel('Value')
    ax.legend(title='Surface')

    # Adding data labels
    for p in ax.patches:
        ax.annotate(f'{p.get_height():.1f}',
                    (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom')

    plt.tight_layout()
    plt.show()
```



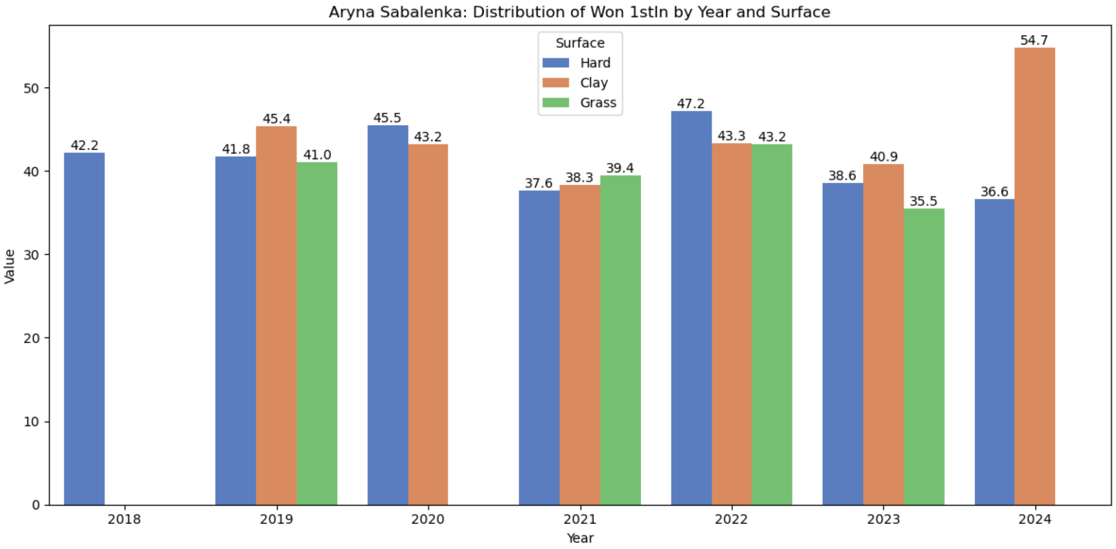

Coco Gauff: Distribution of Loss 1stWon by Year and Surface
Surface
Grass
Hard
Clay
Value
Year
0
5
10
15
20
25
30
35
40
2019
2020
2021
2022
2023
2024
21.0
15.5
29.8
39.0
29.5
32.2
24.2
27.5
28.6
20.2
26.7
25.0
23.8
29.8
33.3

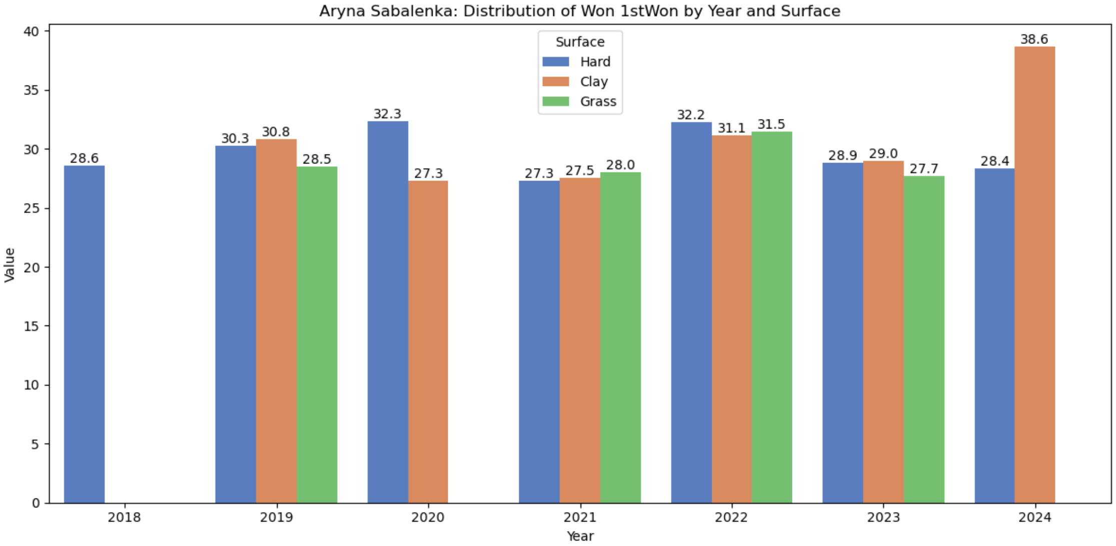

Coco Gauff: Distribution of Loss 2ndWon by Year and Surface
Surface
Grass
Hard
Clay
Value
Year
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
2019
2020
2021
2022
2023
2024
8.0
6.2
12.7
11.5
15.0
15.4
13.5
12.5
11.4
11.5
12.3
11.6
9.0
12.2
21.0

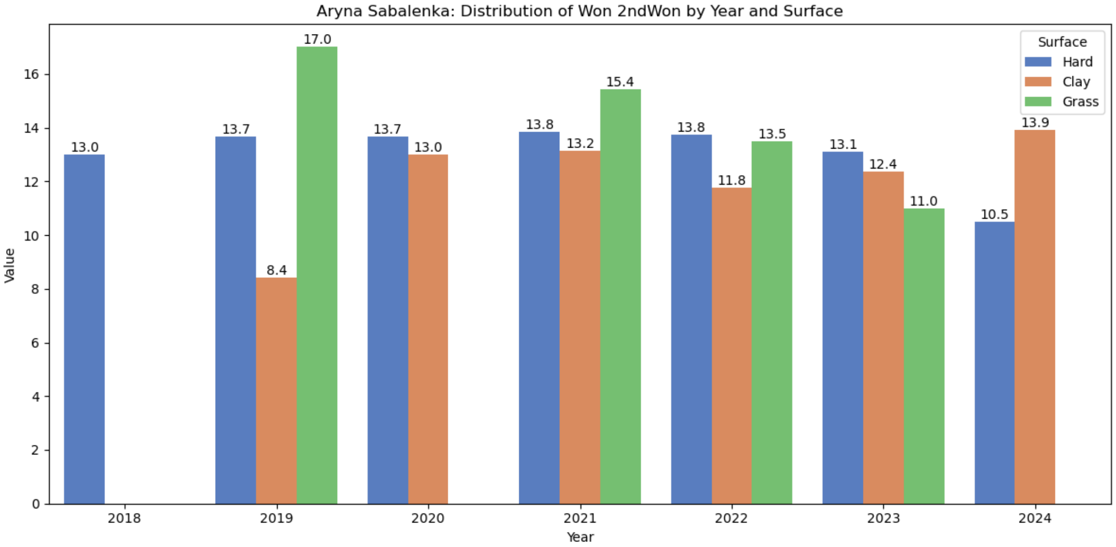


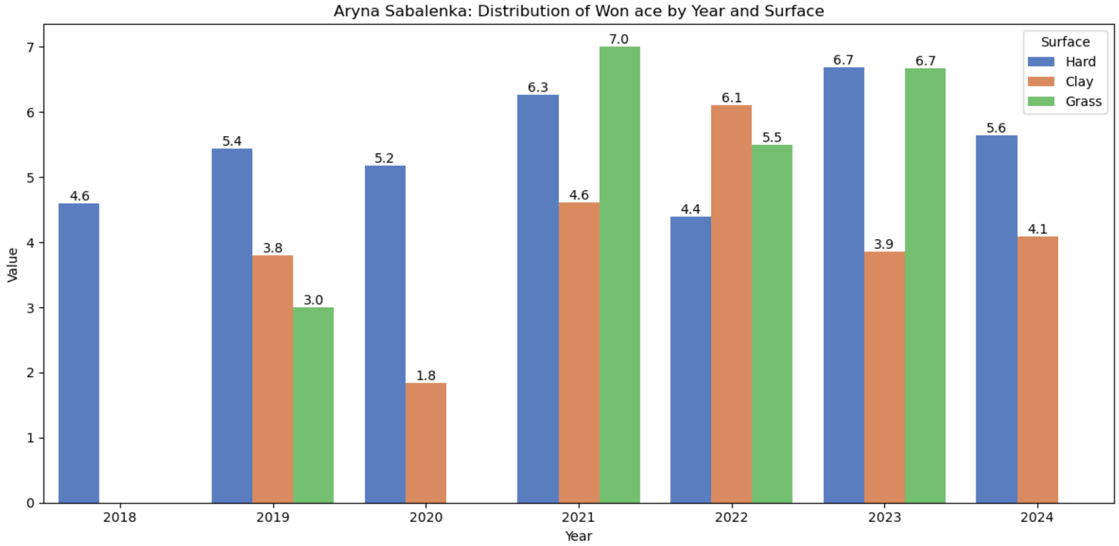


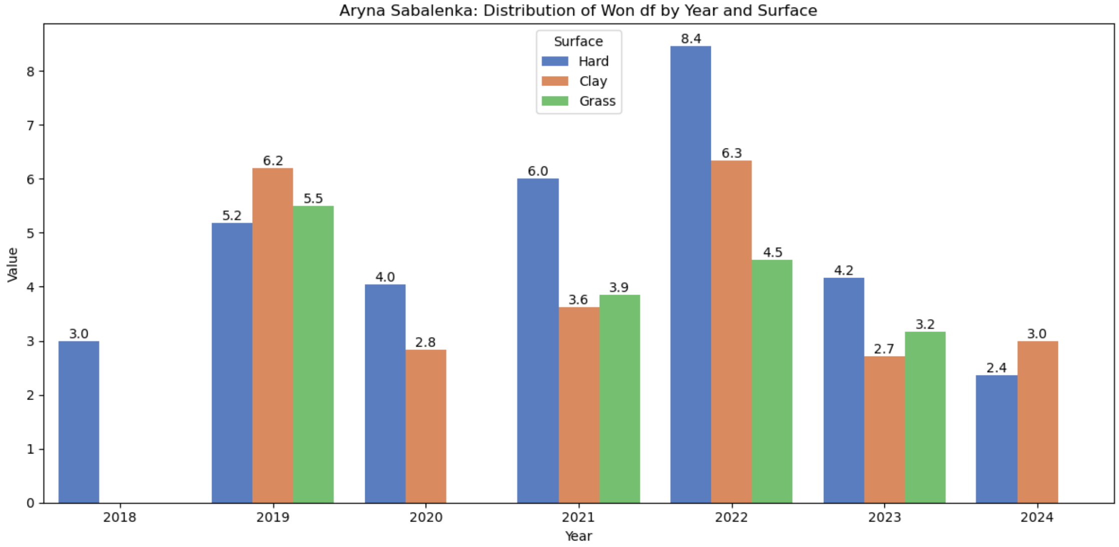
[142]:
```
# Melting the DataFrame for easier plotting
ASw_melted = ASw_summary.melt(id_vars=['Year', 'surface'], value_vars=['1stIn',␣
↪'1stWon', '2ndWon', 'ace', 'df'],
                    var_name='Metric', value_name='Value')

for metric in metrics:
    plt.figure(figsize=(12, 6))
    ax = sns.barplot(data=ASw_melted[ASw_melted['Metric'] == metric], x='Year',␣
 ↪y='Value', hue='surface', errorbar=None, palette='muted')
    ax.set_title(f'Aryna Sabalenka: Distribution of Won {metric} by Year and␣
 ↪Surface')
    ax.set_xlabel('Year')
    ax.set_ylabel('Value')
    ax.legend(title='Surface')

    # Adding data labels
    for p in ax.patches:
        ax.annotate(f'{p.get_height():.1f}',
                    (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom')

    plt.tight_layout()
    plt.show()
```




Aryna Sabalenka: Distribution of Won 1stWon by Year and Surface
Surface
Hard
Clay
Grass
Value
Year
0
5
10
15
20
25
30
35
40
2018
2019
2020
2021
2022
2023
2024
28.6
30.3
30.8
28.5
32.3
27.3
27.3
27.5
28.0
32.2
31.1
31.5
28.9
29.0
27.7
28.4
38.6


Aryna Sabalenka: Distribution of Won 2ndWon by Year and Surface
Surface
Hard
Clay
Grass
Value
Year
0
2
4
6
8
10
12
14
16
2018
2019
2020
2021
2022
2023
2024
13.0
13.7
8.4
17.0
13.7
13.0
13.8
13.2
15.4
13.8
11.8
13.5
13.1
12.4
11.0
10.5
13.9

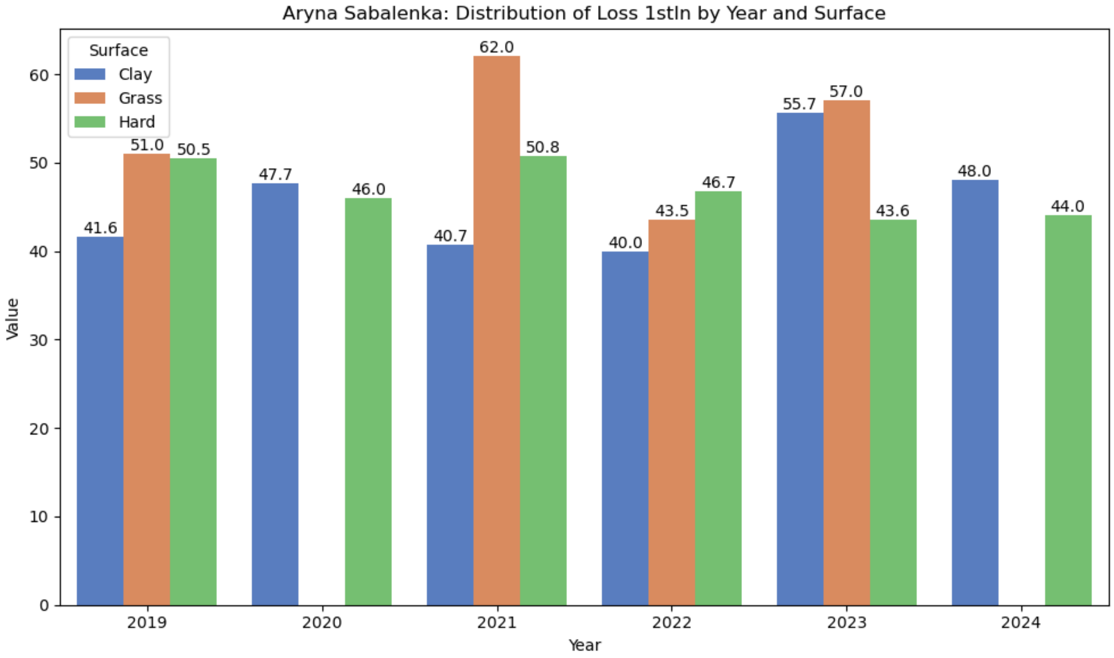


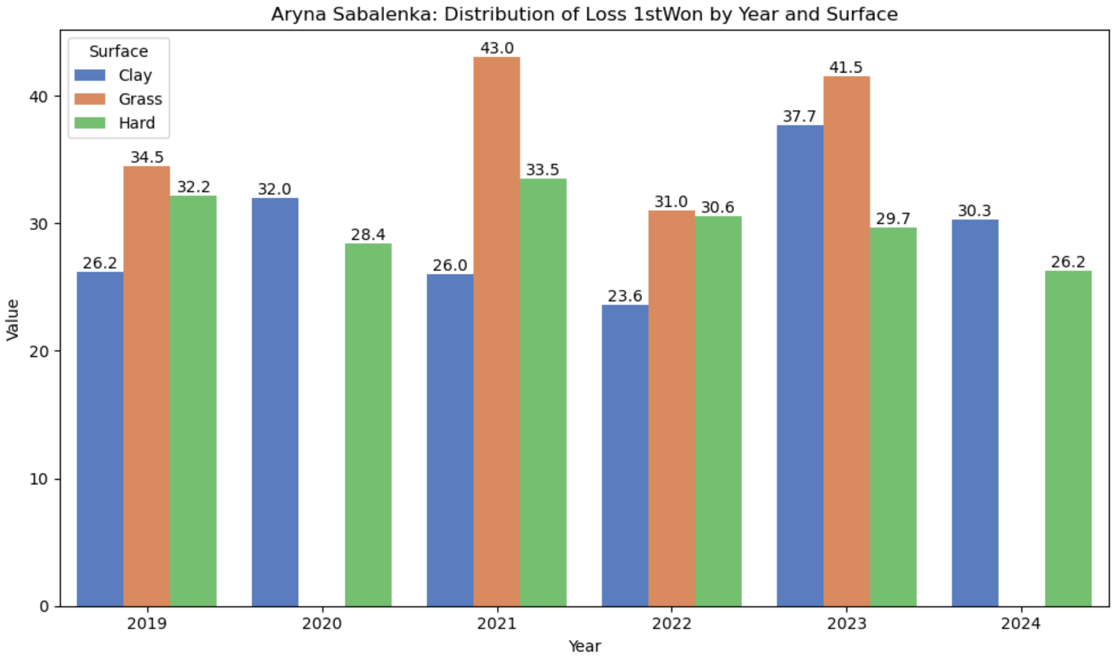


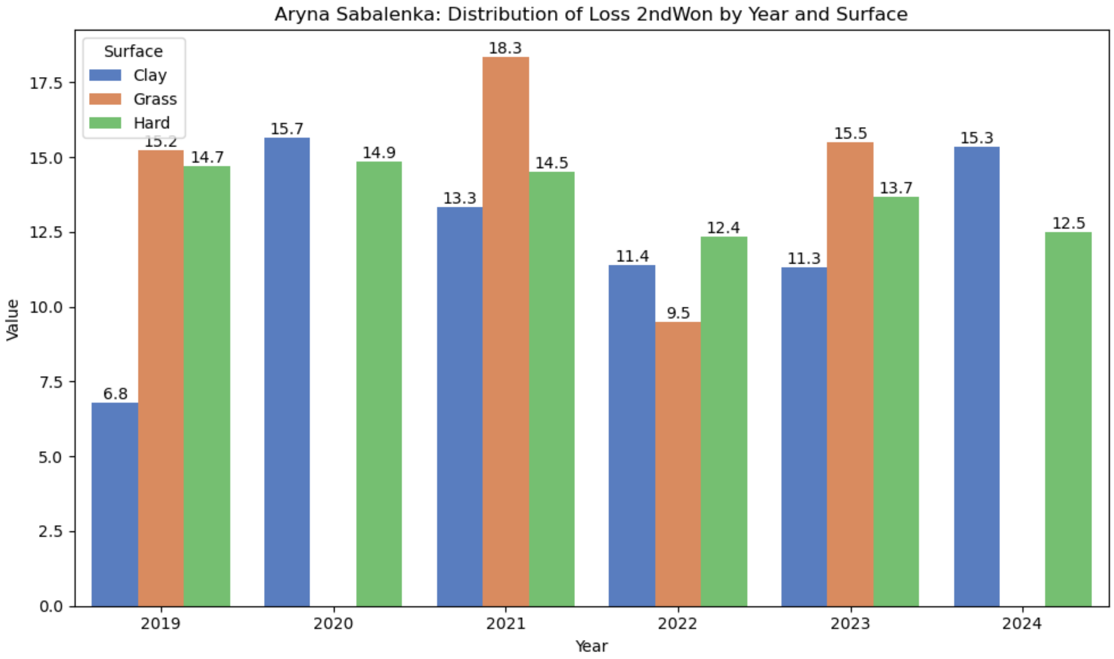
[143]:

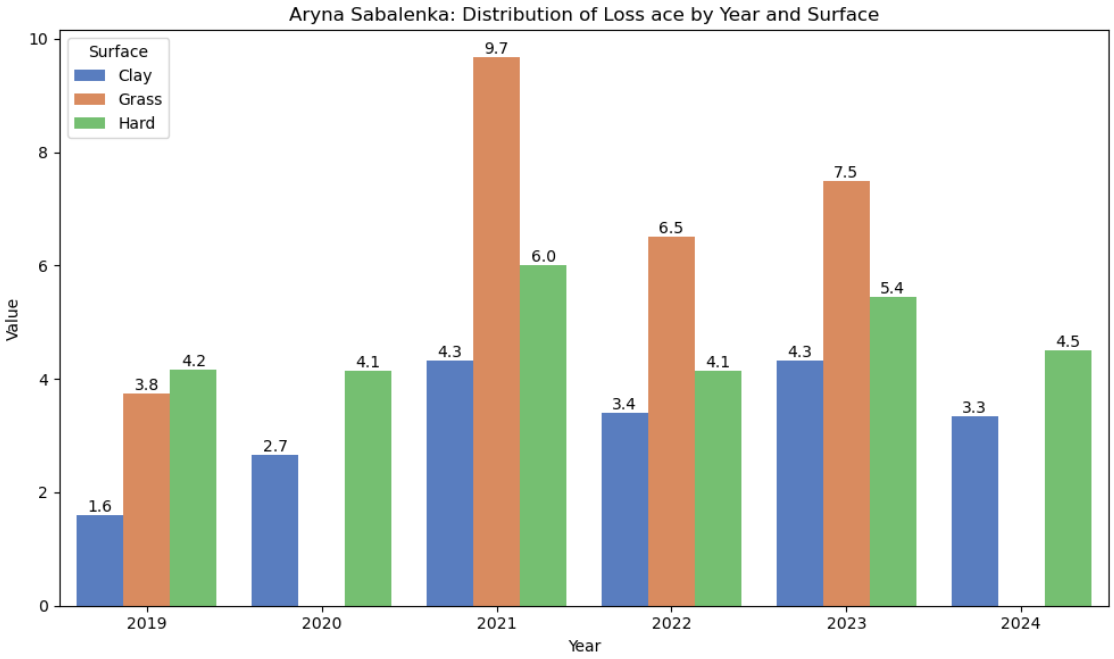
```
# Melting the DataFrame for easier plotting
ASl_melted = ASl_summary.melt(id_vars=['Year', 'surface'], value_vars=['1stIn',␣
↪'1stWon', '2ndWon', 'ace', 'df'],
                    var_name='Metric', value_name='Value')

for metric in metrics:
    plt.figure(figsize=(10, 6))
    ax = sns.barplot(data=ASl_melted[ASl_melted['Metric'] == metric], x='Year',␣
↪y='Value', hue='surface', errorbar=None, palette='muted')
```

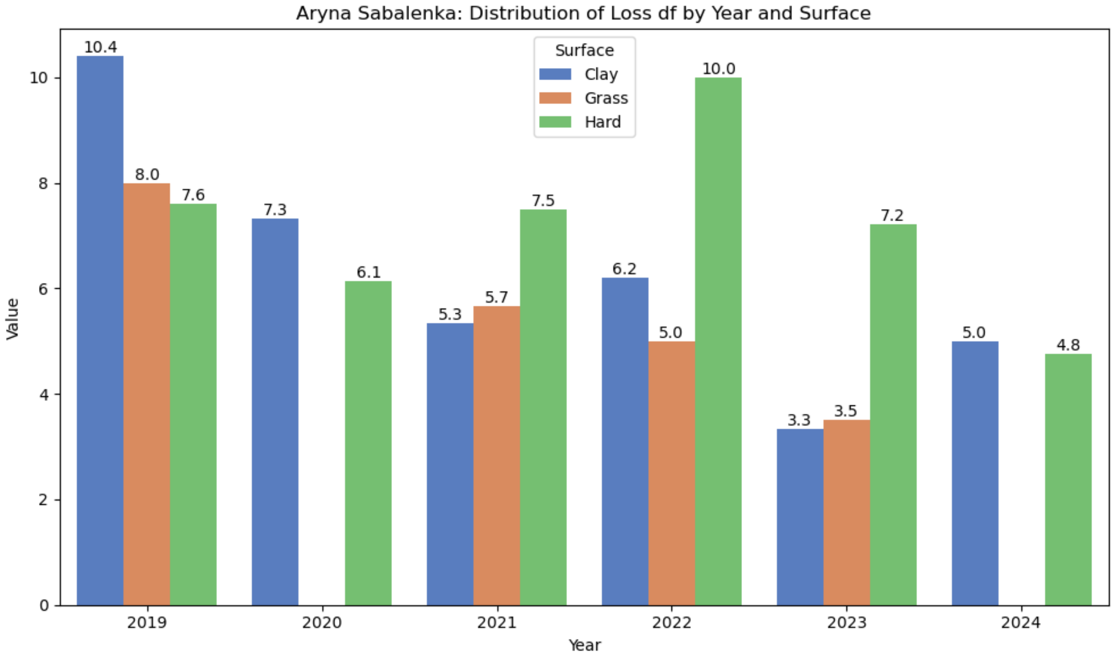
```
    ax.set_title(f'Aryna Sabalenka: Distribution of Loss {metric} by Year and Surface')
    ax.set_xlabel('Year')
    ax.set_ylabel('Value')
    ax.legend(title='Surface')

    # Adding data labels
    for p in ax.patches:
        ax.annotate(f'{p.get_height():.1f}',
                    (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom')

    plt.tight_layout()
    plt.show()
```



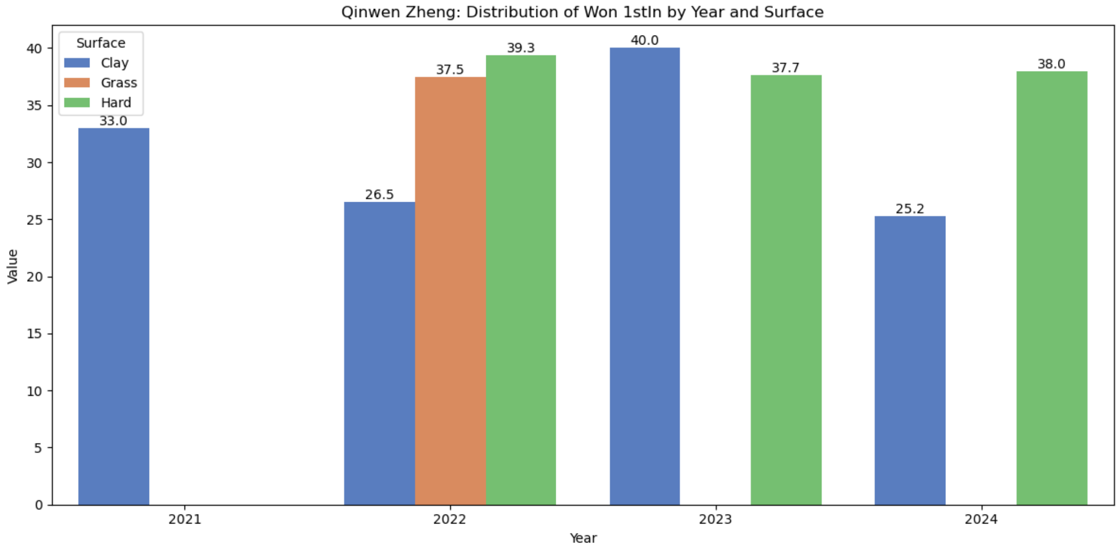

Aryna Sabalenka: Distribution of Loss 1stWon by Year and Surface
Surface
Clay
Grass
Hard
Value
0
10
20
30
40
Year
2019
2020
2021
2022
2023
2024
26.2
34.5
32.2
32.0
28.4
26.0
43.0
33.5
23.6
31.0
30.6
37.7
41.5
29.7
30.3
26.2

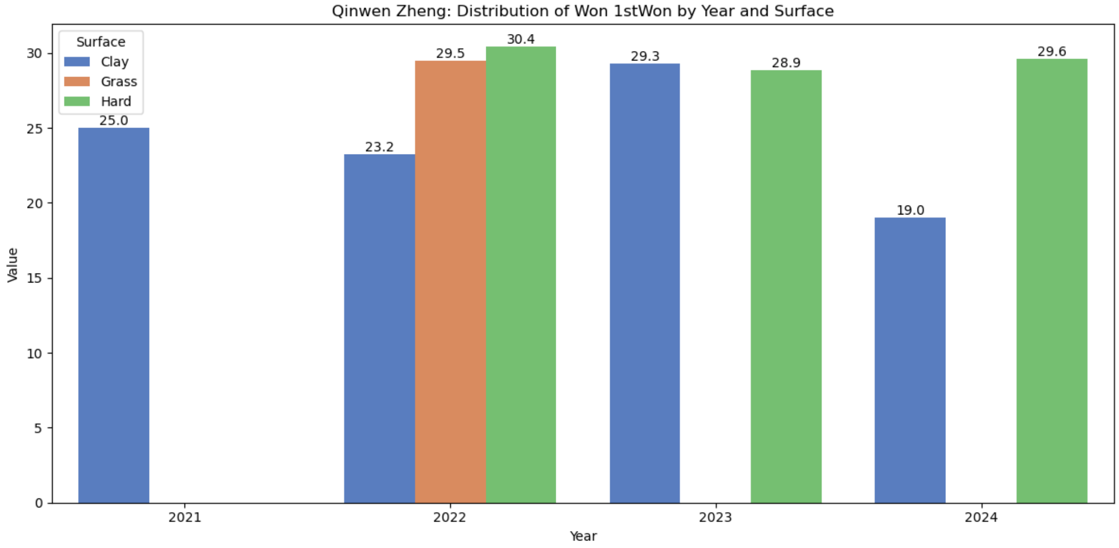

Aryna Sabalenka: Distribution of Loss 2ndWon by Year and Surface
Surface
Clay
Grass
Hard
Value
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Year
2019
2020
2021
2022
2023
2024
6.8
15.2
14.7
15.7
14.9
13.3
18.3
14.5
11.4
9.5
12.4
11.3
15.5
13.7
15.3
12.5

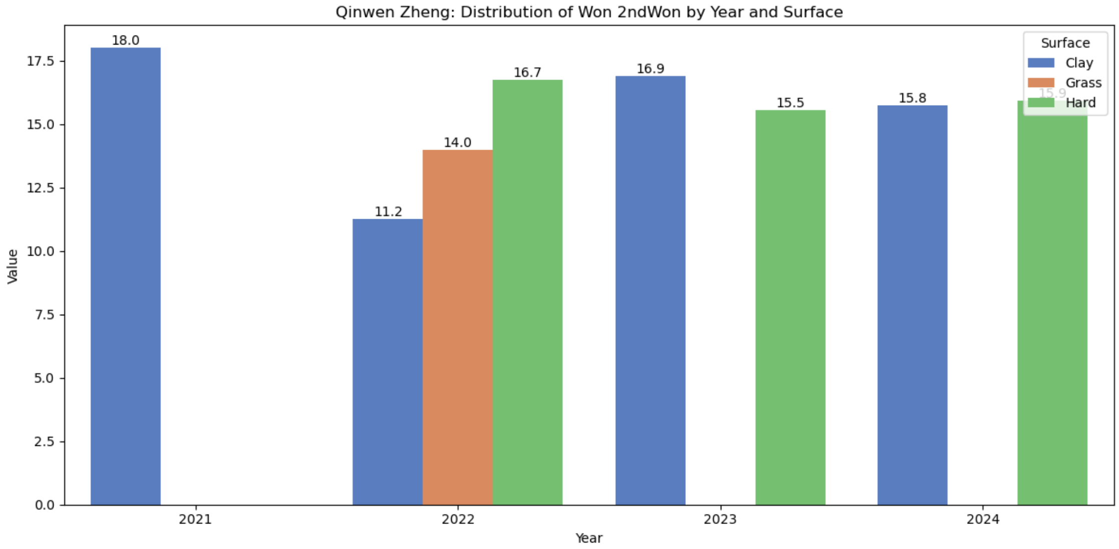


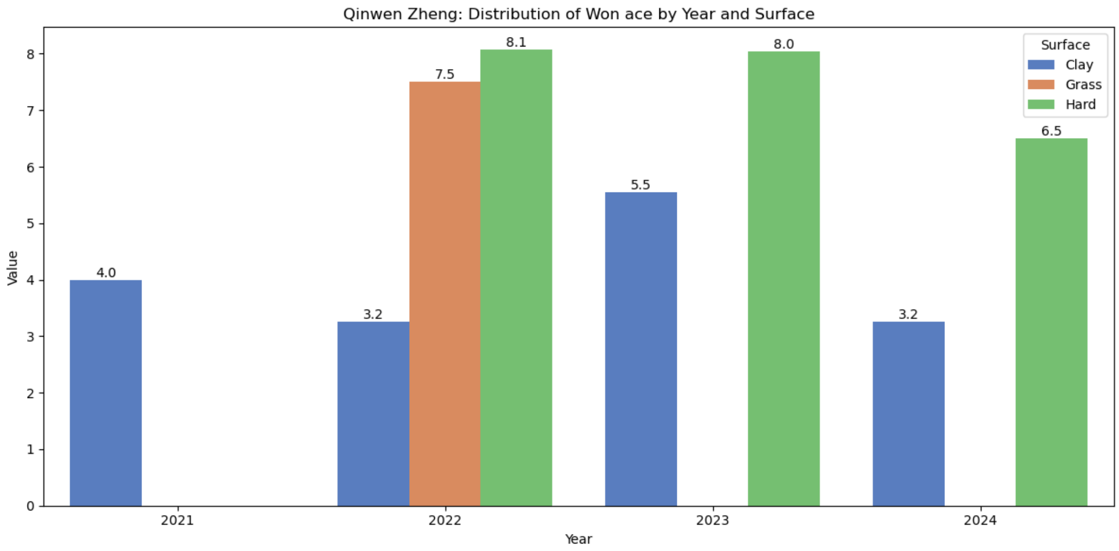


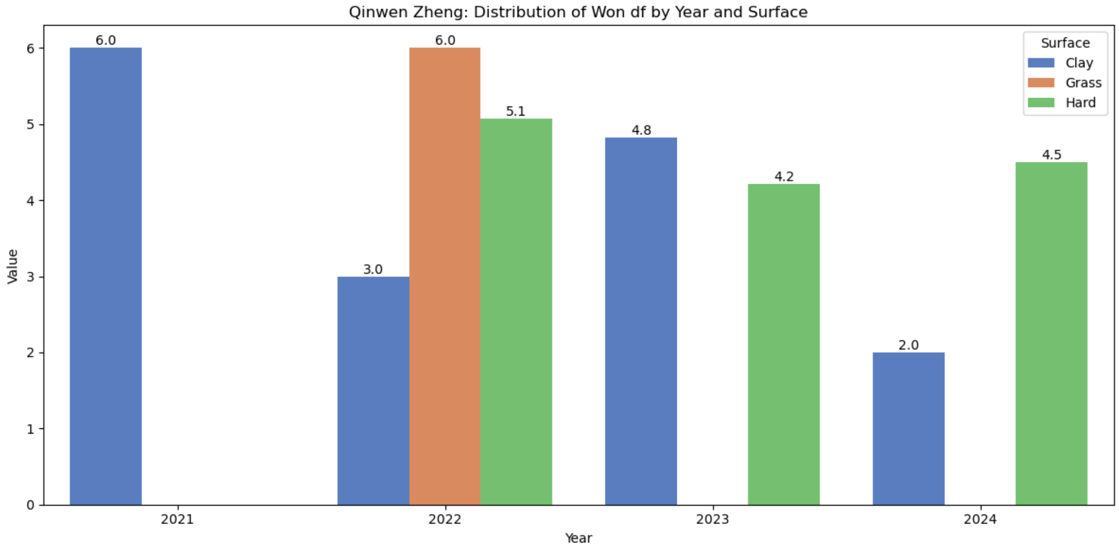
```
[145]: # Melting the DataFrame for easier plotting
QZw_melted = QZw_summary.melt(id_vars=['Year', 'surface'], value_vars=['1stIn',␣
 ↪'1stWon', '2ndWon', 'ace', 'df'],
                        var_name='Metric', value_name='Value')

for metric in metrics:
    plt.figure(figsize=(12, 6))
    ax = sns.barplot(data=QZw_melted[QZw_melted['Metric'] == metric], x='Year',␣
 ↪y='Value', hue='surface', errorbar=None, palette='muted')
    ax.set_title(f'Qinwen Zheng: Distribution of Won {metric} by Year and␣
 ↪Surface')
    ax.set_xlabel('Year')
    ax.set_ylabel('Value')
    ax.legend(title='Surface')

    # Adding data labels
    for p in ax.patches:
        ax.annotate(f'{p.get_height():.1f}',
                    (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom')

    plt.tight_layout()
    plt.show()
```




Qinwen Zheng: Distribution of Won 1stWon by Year and Surface
Surface
Clay
Grass
Hard
Value
Year
0
5
10
15
20
25
30
2021
2022
2023
2024
25.0
23.2
29.5
30.4
29.3
28.9
19.0
29.6


Qinwen Zheng: Distribution of Won 2ndWon by Year and Surface
Surface
Clay
Grass
Hard
Value
Year
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
2021
2022
2023
2024
18.0
11.2
14.0
16.7
16.9
15.5
15.8
15.9

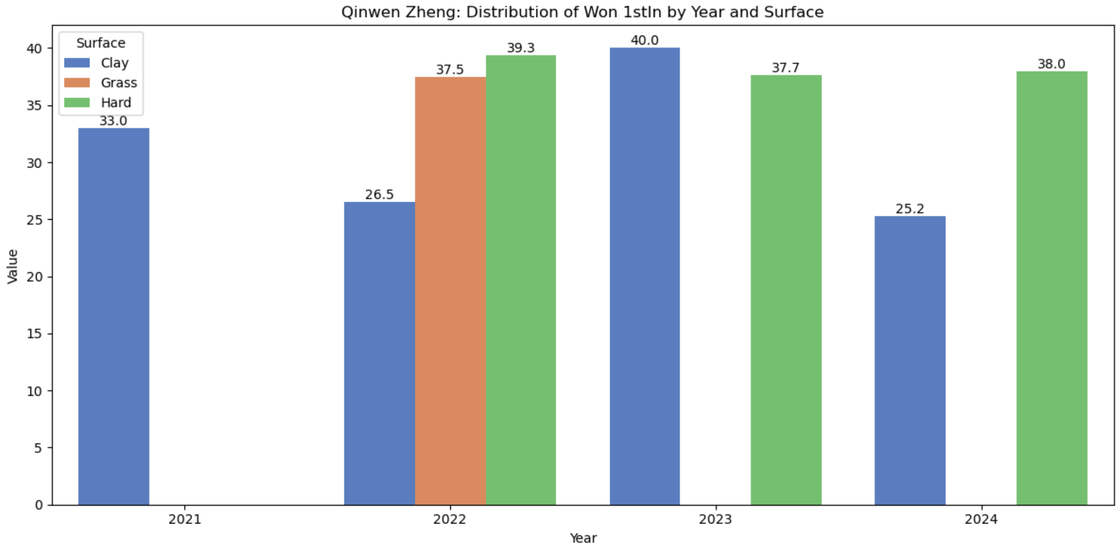


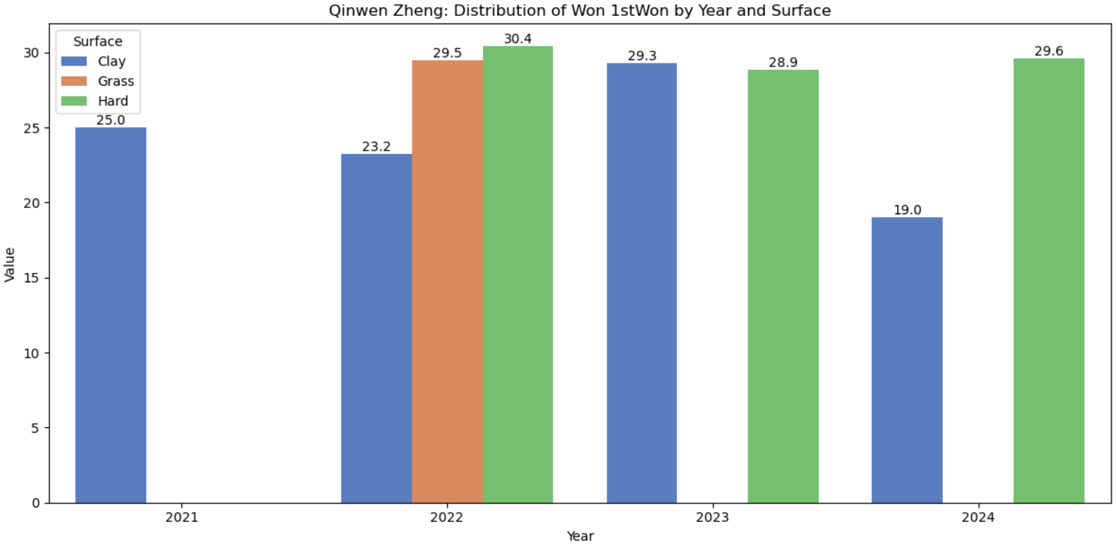


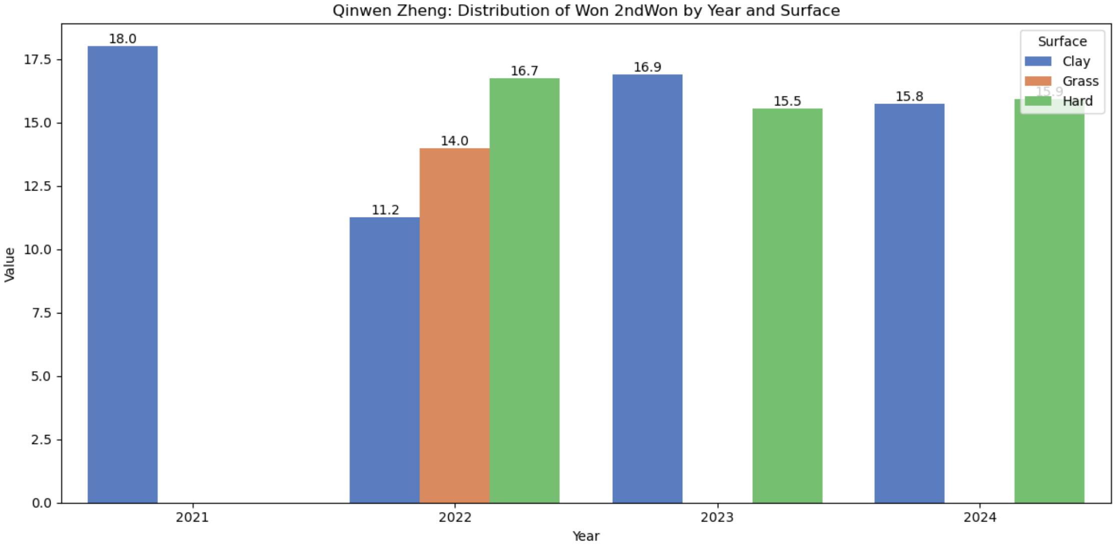
[146]:

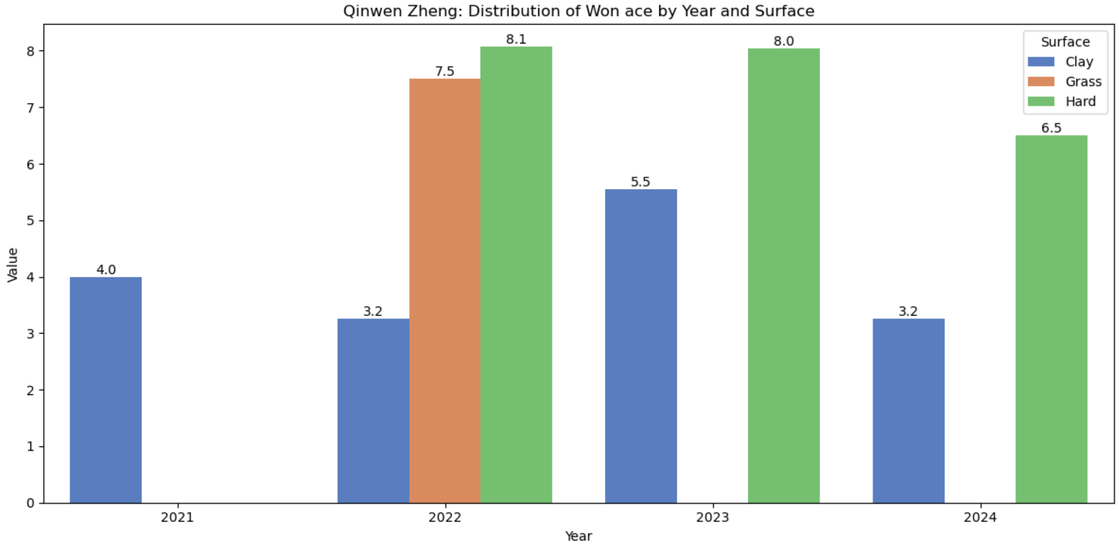
```
# Melting the DataFrame for easier plotting
QZl_melted = QZl_summary.melt(id_vars=['Year', 'surface'], value_vars=['1stIn',␣
↪'1stWon', '2ndWon', 'ace', 'df'],
                    var_name='Metric', value_name='Value')

for metric in metrics:
    plt.figure(figsize=(12, 6))
    ax = sns.barplot(data=QZw_melted[QZw_melted['Metric'] == metric], x='Year',␣
↪y='Value', hue='surface', errorbar=None, palette='muted')
```

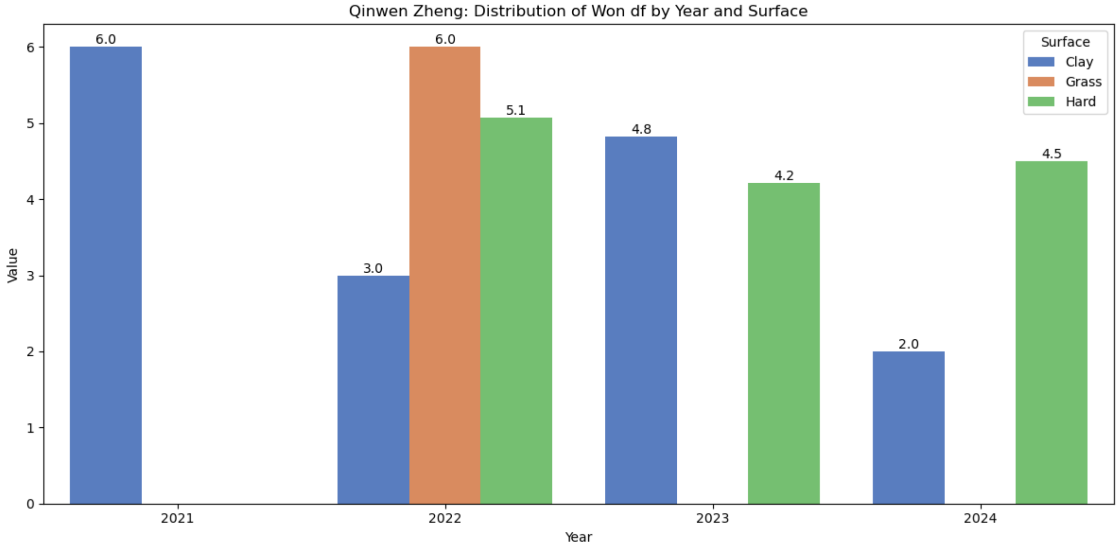
```
    ax.set_title(f'Qinwen Zheng: Distribution of Won {metric} by Year and Surface')
    ax.set_xlabel('Year')
    ax.set_ylabel('Value')
    ax.legend(title='Surface')

    # Adding data labels
    for p in ax.patches:
        ax.annotate(f'{p.get_height():.1f}',
                    (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom')

    plt.tight_layout()
    plt.show()
```




Qinwen Zheng: Distribution of Won 1stWon by Year and Surface
Surface
Clay
Grass
Hard
25.0
23.2
29.5
30.4
29.3
28.9
19.0
29.6
Value
0
5
10
15
20
25
30
2021
2022
2023
2024
Year


Qinwen Zheng: Distribution of Won 2ndWon by Year and Surface
Surface
Clay
Grass
Hard
18.0
11.2
14.0
16.7
16.9
15.5
15.8
15.9
Value
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
2021
2022
2023
2024
Year





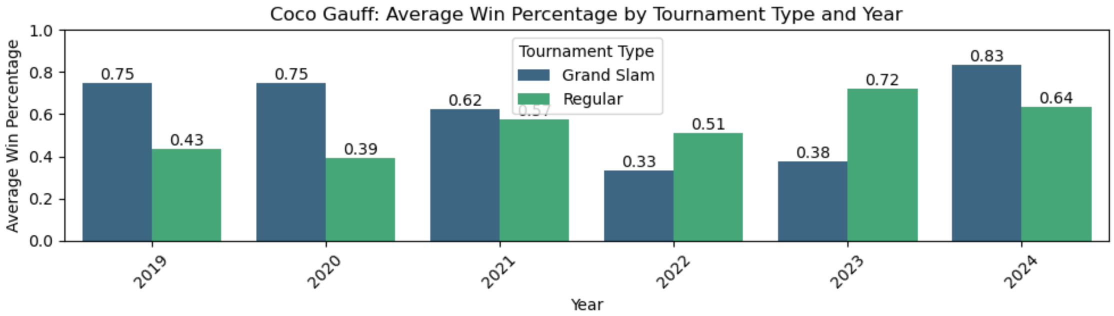
0.7 Against Tournament by Overall Win %

[148]:
```
# Group by Year and Surface, and calculate average win percentage

Cgtourney_performance = Cgyears.groupby(['Year', 'tourney_name']).
↪agg({'Win_Percentage': 'mean'}).reset_index()

Cgtourney_performance['Type'] = np.where(
    Cgtourney_performance['tourney_name'].isin(['US Open', 'Australian Open',␣
↪'Wimbledon']),
    'Grand Slam',
    'Regular'
)

Astourney_performance = Asyears.groupby(['Year', 'tourney_name']).
 ↪agg({'Win_Percentage': 'mean'}).reset_index()

Astourney_performance['Type'] = np.where(
    Astourney_performance['tourney_name'].isin(['US Open', 'Australian Open',␣
 ↪'Wimbledon']),
    'Grand Slam',
    'Regular'
)

Qztourney_performance = qzyears.groupby(['Year', 'tourney_name']).
 ↪agg({'Win_Percentage': 'mean'}).reset_index()

Qztourney_performance['Type'] = np.where(
    Qztourney_performance['tourney_name'].isin(['US Open', 'Australian Open',␣
 ↪'Wimbledon']),
    'Grand Slam',
    'Regular'
)
```

[159]:

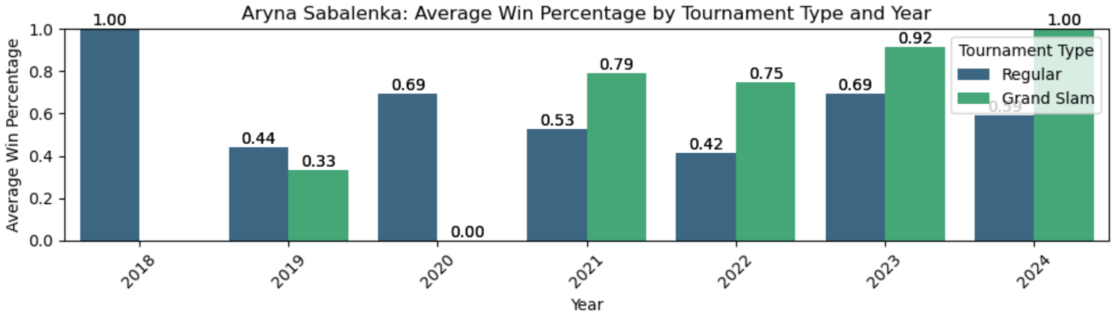
```
# Calculate average Win_Percentage by Year and Type
Cgaverage_win_percentage = Cgtourney_performance.groupby(['Year',␣
 ↪'Type'])['Win_Percentage'].mean().reset_index()

# Plotting
plt.figure(figsize=(10, 3))
sns.barplot(data=Cgaverage_win_percentage, x='Year', y='Win_Percentage',␣
 ↪hue='Type', palette='viridis')

# Customize the plot
plt.title('Coco Gauff: Average Win Percentage by Tournament Type and Year')
plt.xlabel('Year')
plt.ylabel('Average Win Percentage')
plt.legend(title='Tournament Type')
plt.ylim(0, 1)  # Set y-axis limits from 0 to 1
plt.xticks(rotation=45)
plt.tight_layout()

# Adding data labels
for p in plt.gca().patches:
    plt.annotate(f'{p.get_height():.2f}',
                 (p.get_x() + p.get_width() / 2., p.get_height()),
                 ha='center', va='bottom')
```

```
plt.tight_layout()

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



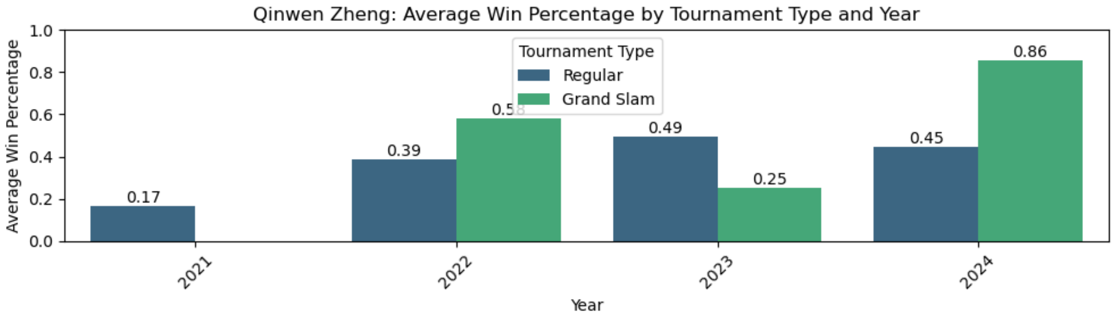
[158]:

```
# Calculate average Win_Percentage by Year and Type
Asaverage_win_percentage = Astourney_performance.groupby(['Year',␣
 ↪'Type'])['Win_Percentage'].mean().reset_index()

# Plotting
plt.figure(figsize=(10, 3))
sns.barplot(data=Asaverage_win_percentage, x='Year', y='Win_Percentage',␣
 ↪hue='Type', palette='viridis')

# Customize the plot
plt.title('Aryna Sabalenka: Average Win Percentage by Tournament Type and Year')
plt.xlabel('Year')
plt.ylabel('Average Win Percentage')
plt.legend(title='Tournament Type')
plt.ylim(0, 1)  # Set y-axis limits from 0 to 1
plt.xticks(rotation=45)
plt.tight_layout()

# Adding data labels
for p in plt.gca().patches:
    plt.annotate(f'{p.get_height():.2f}',
                 (p.get_x() + p.get_width() / 2., p.get_height()),
                 ha='center', va='bottom')

plt.tight_layout()

# Adding data labels
for p in plt.gca().patches:
    plt.annotate(f'{p.get_height():.2f}',
                 (p.get_x() + p.get_width() / 2., p.get_height()),
                 ha='center', va='bottom')

plt.tight_layout()

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



[156]:
```
# Calculate average Win_Percentage by Year and Type
Qzaverage_win_percentage = Qztourney_performance.groupby(['Year',␣
 ↪'Type'])['Win_Percentage'].mean().reset_index()

# Plotting
plt.figure(figsize=(10, 3))
sns.barplot(data=Qzaverage_win_percentage, x='Year', y='Win_Percentage',␣
 ↪hue='Type', palette='viridis')

# Customize the plot
plt.title('Qinwen Zheng: Average Win Percentage by Tournament Type and Year')
plt.xlabel('Year')
plt.ylabel('Average Win Percentage')
plt.legend(title='Tournament Type')
plt.ylim(0, 1)  # Set y-axis limits from 0 to 1
plt.xticks(rotation=45)
plt.tight_layout()

# Adding data labels
for p in plt.gca().patches:
    plt.annotate(f'{p.get_height():.2f}',
                 (p.get_x() + p.get_width() / 2., p.get_height()),
                 ha='center', va='bottom')

plt.tight_layout()
```

```

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



0.8 Against Tournamet by Serve %

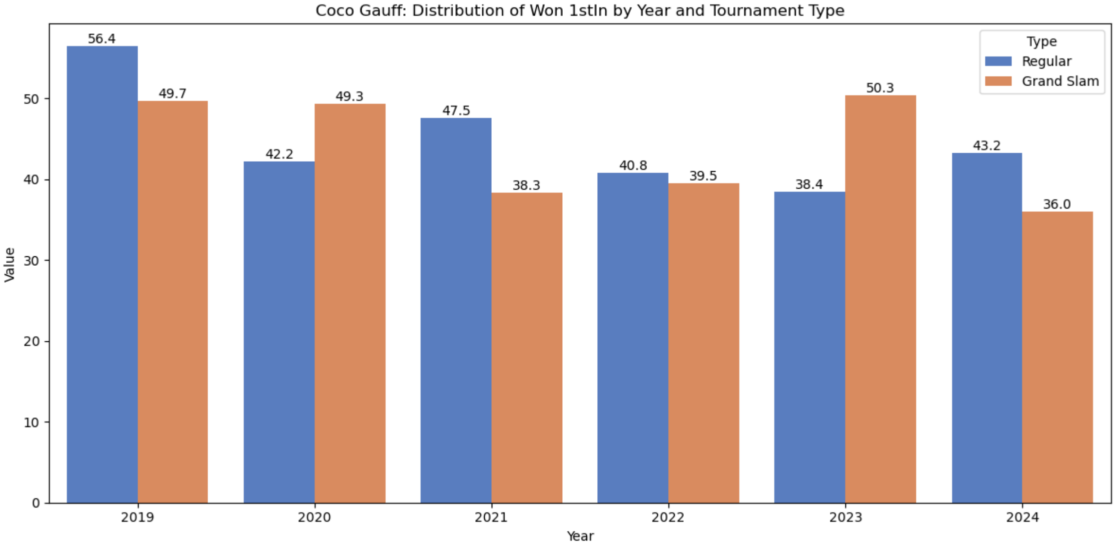
[153]:
```
# Group by Year and Surface, and calculate average win percentage

CGw_summary = CGw.groupby(['Year', 'tourney_name']).agg({'w_1stIn': 'mean',␣
 ↪'w_1stWon': 'mean', 'w_2ndWon': 'mean', 'w_ace': 'mean', 'w_df': 'mean'}).
 ↪reset_index()

CGw_summary['Type'] = np.where(
    CGw_summary['tourney_name'].isin(['US Open', 'Australian Open',␣
 ↪'Wimbledon']),
    'Grand Slam',
    'Regular'
)

CGw_summary.rename({'w_1stIn': '1stIn', 'w_1stWon': '1stWon', 'w_2ndWon':␣
 ↪'2ndWon', 'w_ace': 'ace', 'w_df': 'df'}, axis=1, inplace=True)
CGl_summary = CGl.groupby(['Year', 'tourney_name']).agg({'l_1stIn': 'mean',␣
 ↪'l_1stWon': 'mean', 'l_2ndWon': 'mean', 'l_ace': 'mean', 'l_df': 'mean'}).
 ↪reset_index()

CGl_summary['Type'] = np.where(
    CGl_summary['tourney_name'].isin(['US Open', 'Australian Open',␣
 ↪'Wimbledon']),
    'Grand Slam',
    'Regular'
)

CGl_summary.rename({'l_1stIn': '1stIn', 'l_1stWon': '1stWon', 'l_2ndWon':␣
 ↪'2ndWon', 'l_ace': 'ace', 'l_df': 'df'}, axis=1, inplace=True)
```

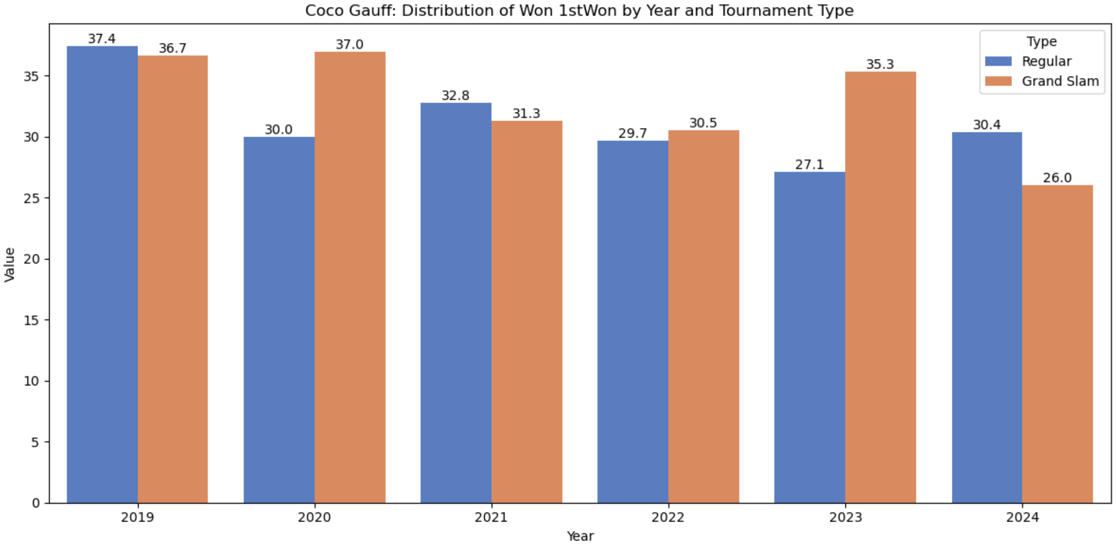
```
# Group by Year and Surface, and calculate average win percentage

ASw_summary = ASw.groupby(['Year', 'tourney_name']).agg({'w_1stIn': 'mean',␣
 ↪'w_1stWon': 'mean', 'w_2ndWon': 'mean', 'w_ace': 'mean', 'w_df': 'mean'}).
 ↪reset_index()

ASw_summary['Type'] = np.where(
    ASw_summary['tourney_name'].isin(['US Open', 'Australian Open',␣
 ↪'Wimbledon']),
    'Grand Slam',
    'Regular'
)

ASw_summary.rename({'w_1stIn': '1stIn', 'w_1stWon': '1stWon', 'w_2ndWon':␣
 ↪'2ndWon', 'w_ace': 'ace', 'w_df': 'df'}, axis=1, inplace=True)
ASl_summary = ASl.groupby(['Year', 'tourney_name']).agg({'l_1stIn': 'mean',␣
 ↪'l_1stWon': 'mean', 'l_2ndWon': 'mean', 'l_ace': 'mean', 'l_df': 'mean'}).
 ↪reset_index()

ASl_summary['Type'] = np.where(
    ASl_summary['tourney_name'].isin(['US Open', 'Australian Open',␣
 ↪'Wimbledon']),
    'Grand Slam',
    'Regular'
)

ASl_summary.rename({'l_1stIn': '1stIn', 'l_1stWon': '1stWon', 'l_2ndWon':␣
 ↪'2ndWon', 'l_ace': 'ace', 'l_df': 'df'}, axis=1, inplace=True)

# Group by Year and Surface, and calculate average win percentage

QZw_summary = QZw.groupby(['Year', 'tourney_name']).agg({'w_1stIn': 'mean',␣
 ↪'w_1stWon': 'mean', 'w_2ndWon': 'mean', 'w_ace': 'mean', 'w_df': 'mean'}).
 ↪reset_index()

QZw_summary['Type'] = np.where(
    QZw_summary['tourney_name'].isin(['US Open', 'Australian Open',␣
 ↪'Wimbledon']),
    'Grand Slam',
    'Regular'
)

QZw_summary.rename({'w_1stIn': '1stIn', 'w_1stWon': '1stWon', 'w_2ndWon':␣
 ↪'2ndWon', 'w_ace': 'ace', 'w_df': 'df'}, axis=1, inplace=True)
```

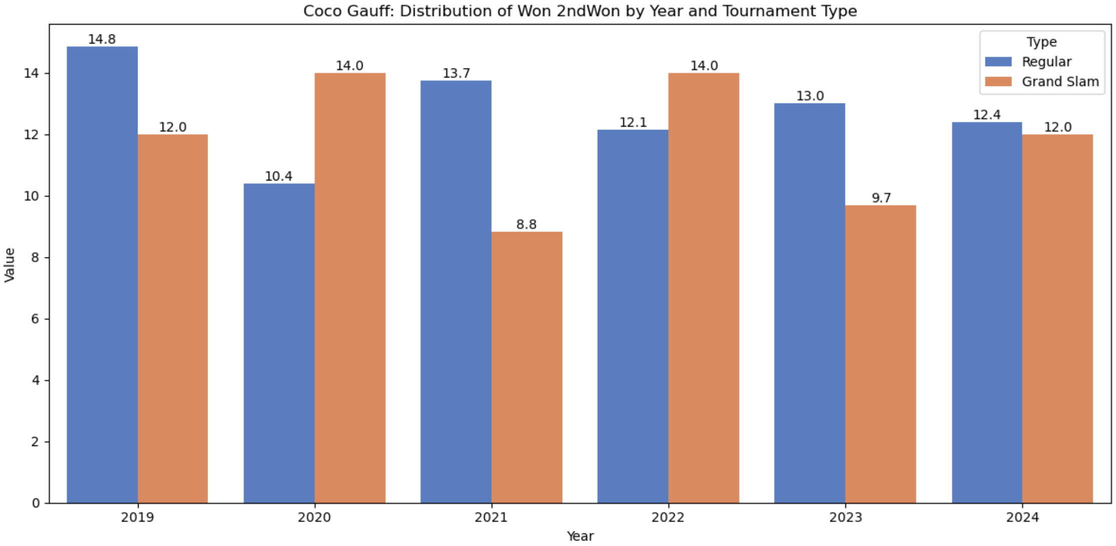
```
QZl_summary = QZl.groupby(['Year', 'tourney_name']).agg({'l_1stIn': 'mean',␣
 ↪'l_1stWon': 'mean', 'l_2ndWon': 'mean', 'l_ace': 'mean', 'l_df': 'mean'}).
 ↪reset_index()

QZl_summary['Type'] = np.where(
    QZl_summary['tourney_name'].isin(['US Open', 'Australian Open',␣
 ↪'Wimbledon']),
    'Grand Slam',
    'Regular'
)

QZl_summary.rename({'l_1stIn': '1stIn', 'l_1stWon': '1stWon', 'l_2ndWon':␣
 ↪'2ndWon', 'l_ace': 'ace', 'l_df': 'df'}, axis=1, inplace=True)
```

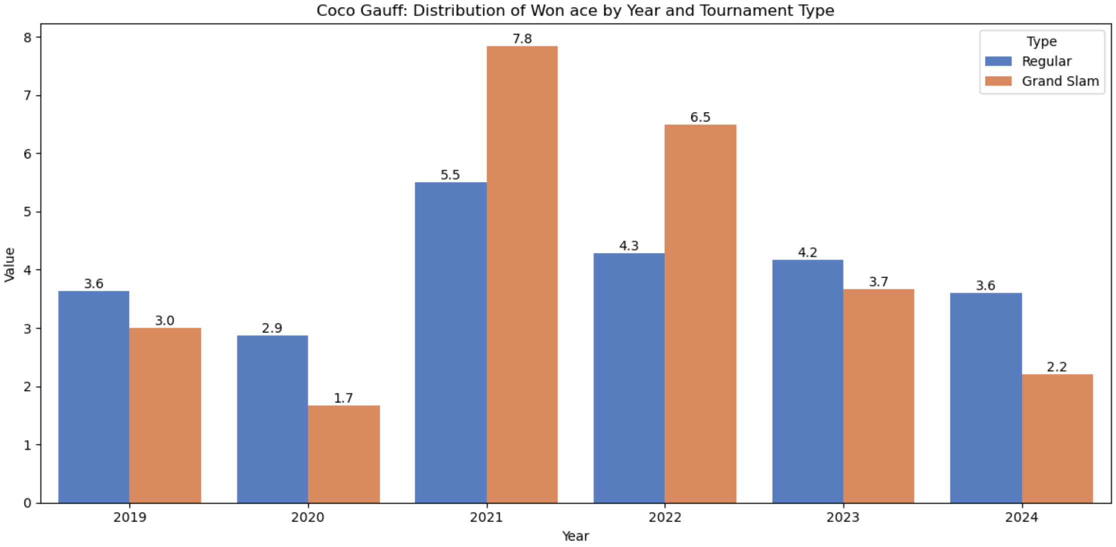
[38]:

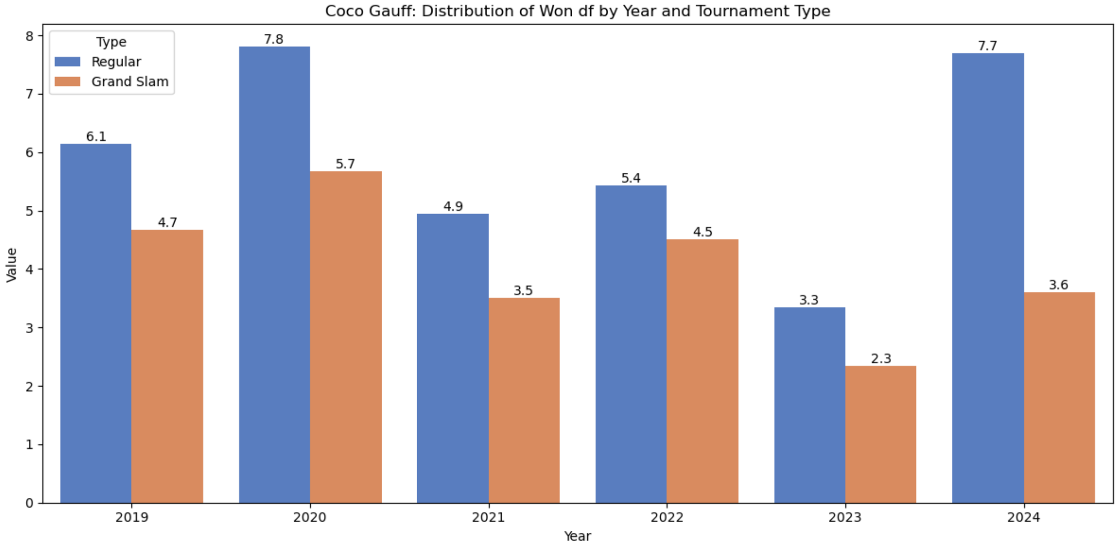
```
# Melting the DataFrame for easier plotting
CGw_melted = CGw_summary.melt(id_vars=['Year', 'Type'], value_vars=['1stIn',␣
 ↪'1stWon', '2ndWon', 'ace', 'df'],
                    var_name='Metric', value_name='Value')

for metric in metrics:
    plt.figure(figsize=(12, 6))
    ax = sns.barplot(data=CGw_melted[CGw_melted['Metric'] == metric], x='Year',␣
 ↪y='Value', hue='Type', errorbar=None, palette='muted')
    ax.set_title(f'Coco Gauff: Distribution of Won {metric} by Year and␣
 ↪Tournament Type')
    ax.set_xlabel('Year')
    ax.set_ylabel('Value')
    ax.legend(title='Type')

    # Adding data labels
    for p in ax.patches:
        ax.annotate(f'{p.get_height():.1f}',
                    (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom')

    plt.tight_layout()
    plt.show()
```


Coco Gauff: Distribution of Won 1stIn by Year and Tournament Type
Type
Regular
Grand Slam
56.4
49.7
42.2
49.3
47.5
38.3
40.8
39.5
38.4
50.3
43.2
36.0
Value
0
10
20
30
40
50
2019
2020
2021
2022
2023
2024
Year

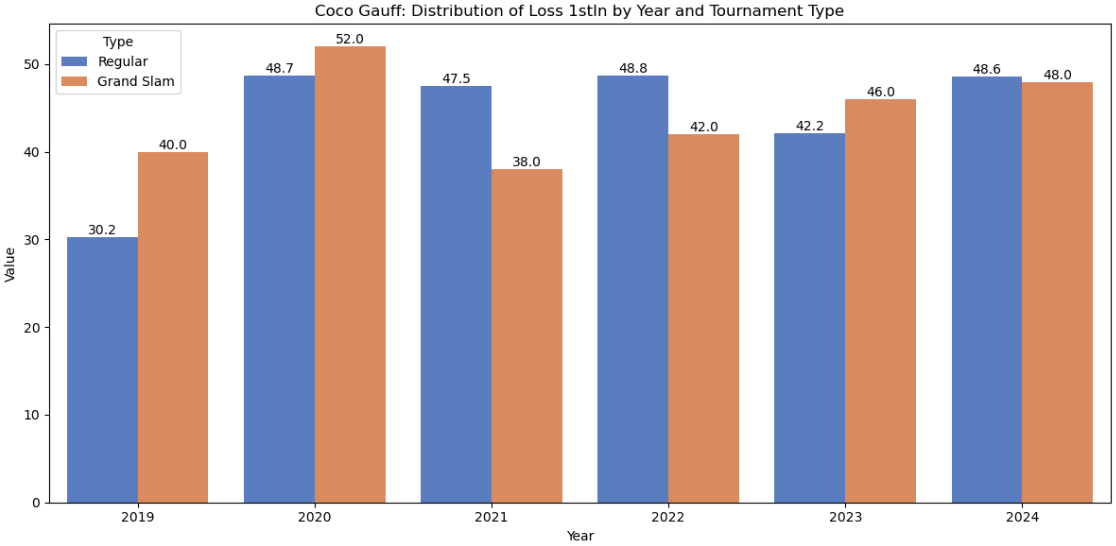

Coco Gauff: Distribution of Won 1stWon by Year and Tournament Type
Type
Regular
Grand Slam
37.4
36.7
30.0
37.0
32.8
31.3
29.7
30.5
27.1
35.3
30.4
26.0
Value
0
5
10
15
20
25
30
35
2019
2020
2021
2022
2023
2024
Year

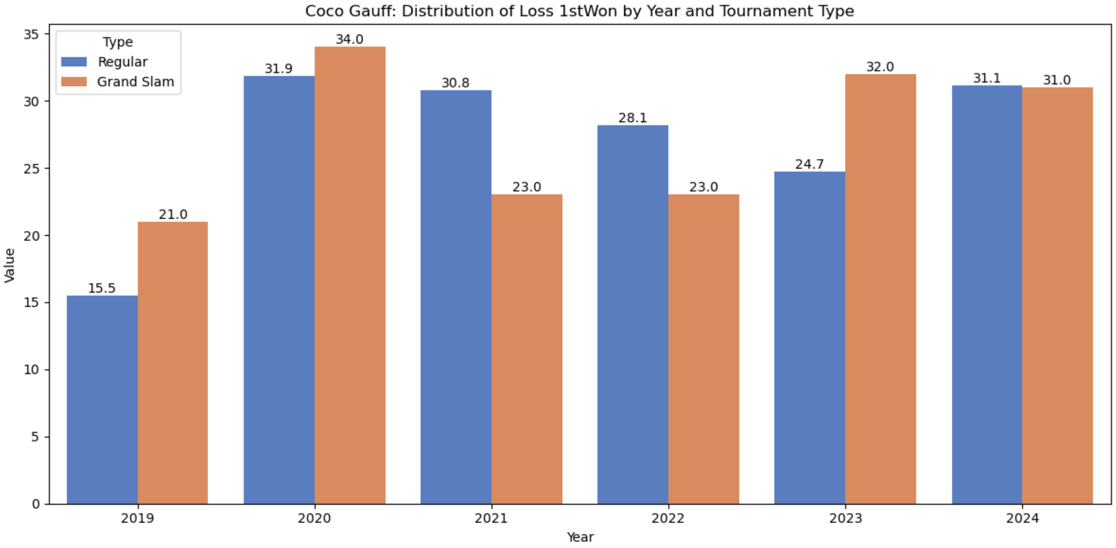

Coco Gauff: Distribution of Won 2ndWon by Year and Tournament Type
Type
Regular
Grand Slam
Value
Year
0
2
4
6
8
10
12
14
2019
2020
2021
2022
2023
2024
14.8
12.0
10.4
14.0
13.7
8.8
12.1
14.0
13.0
9.7
12.4
12.0

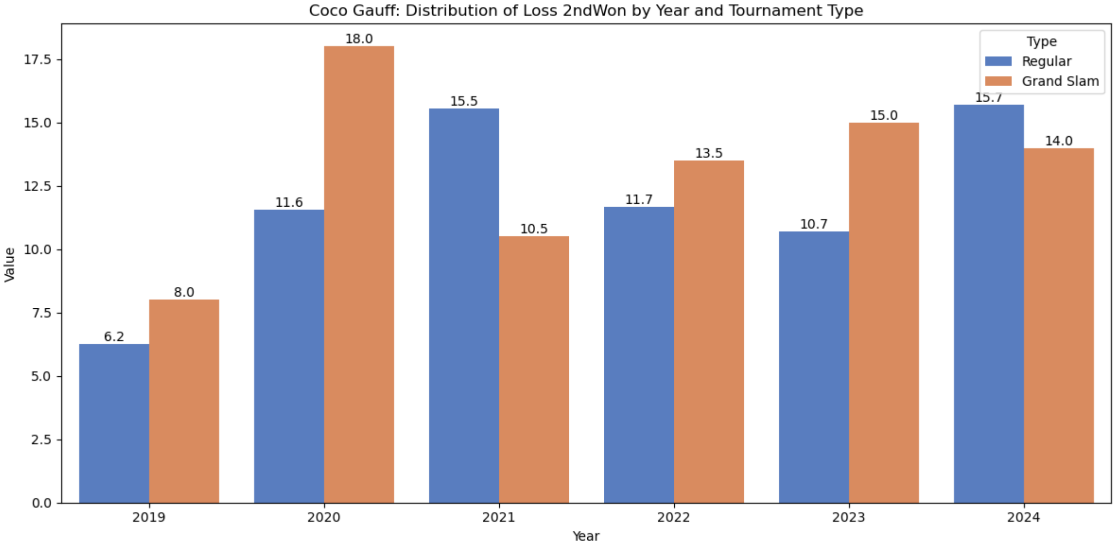

Coco Gauff: Distribution of Won ace by Year and Tournament Type
Type
Regular
Grand Slam
Value
Year
0
1
2
3
4
5
6
7
8
2019
2020
2021
2022
2023
2024
3.6
3.0
2.9
1.7
5.5
7.8
4.3
6.5
4.2
3.7
3.6
2.2

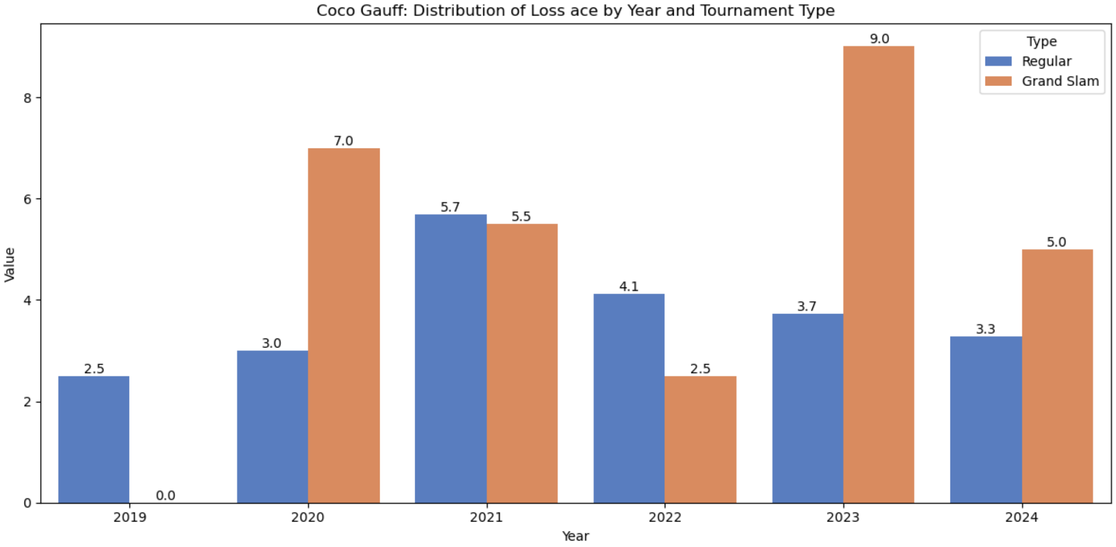


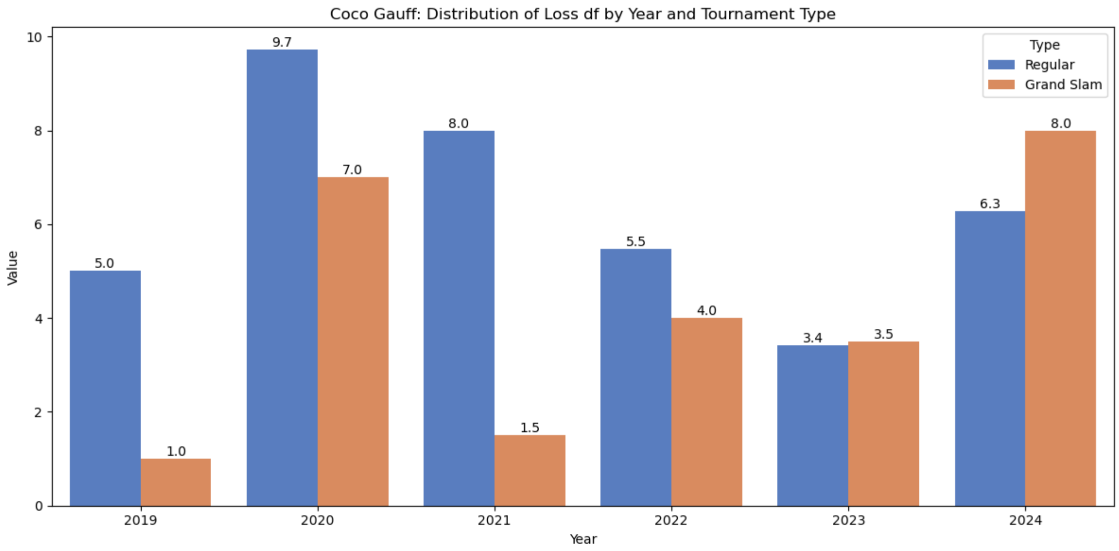
[39]:
```
# Melting the DataFrame for easier plotting
CGl_melted = CGl_summary.melt(id_vars=['Year', 'Type'], value_vars=['1stIn',␣
 ↪'1stWon', '2ndWon', 'ace', 'df'],
                    var_name='Metric', value_name='Value')

for metric in metrics:
    plt.figure(figsize=(12, 6))
    ax = sns.barplot(data=CGl_melted[CGl_melted['Metric'] == metric], x='Year',␣
 ↪y='Value', hue='Type', errorbar=None, palette='muted')
    ax.set_title(f'Coco Gauff: Distribution of Loss {metric} by Year and␣
 ↪Tournament Type')
    ax.set_xlabel('Year')
    ax.set_ylabel('Value')
    ax.legend(title='Type')

    # Adding data labels
    for p in ax.patches:
        ax.annotate(f'{p.get_height():.1f}',
                    (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom')

    plt.tight_layout()
    plt.show()
```


Coco Gauff: Distribution of Loss 1stIn by Year and Tournament Type

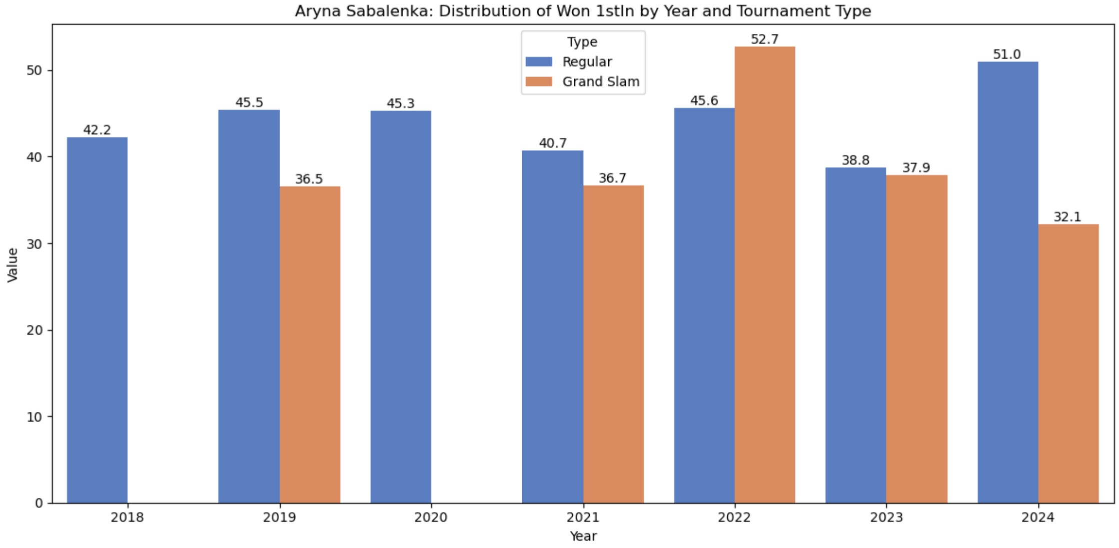

Coco Gauff: Distribution of Loss 1stWon by Year and Tournament Type

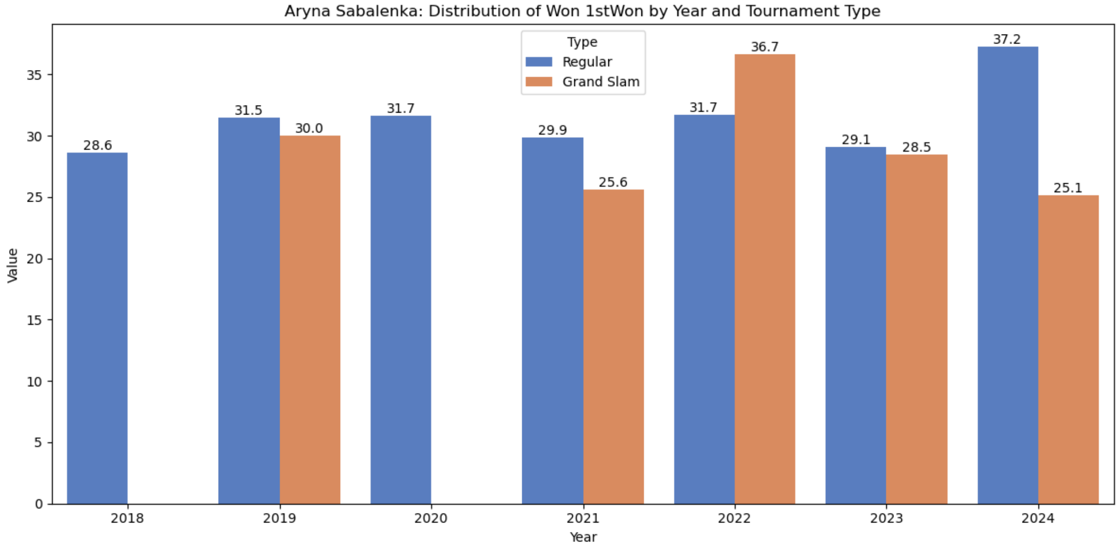

Coco Gauff: Distribution of Loss 2ndWon by Year and Tournament Type
Type
Regular
Grand Slam
Value
Year
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
2019
2020
2021
2022
2023
2024
6.2
8.0
11.6
18.0
15.5
10.5
11.7
13.5
10.7
15.0
15.7
14.0

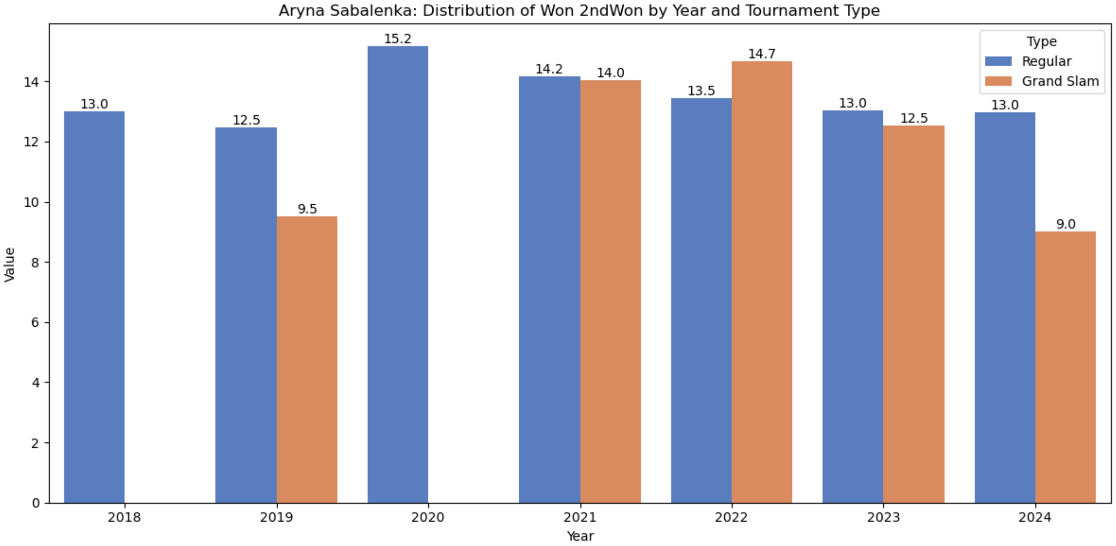

Coco Gauff: Distribution of Loss ace by Year and Tournament Type
Type
Regular
Grand Slam
Value
Year
0
2
4
6
8
2019
2020
2021
2022
2023
2024
2.5
0.0
3.0
7.0
5.7
5.5
4.1
2.5
3.7
9.0
3.3
5.0

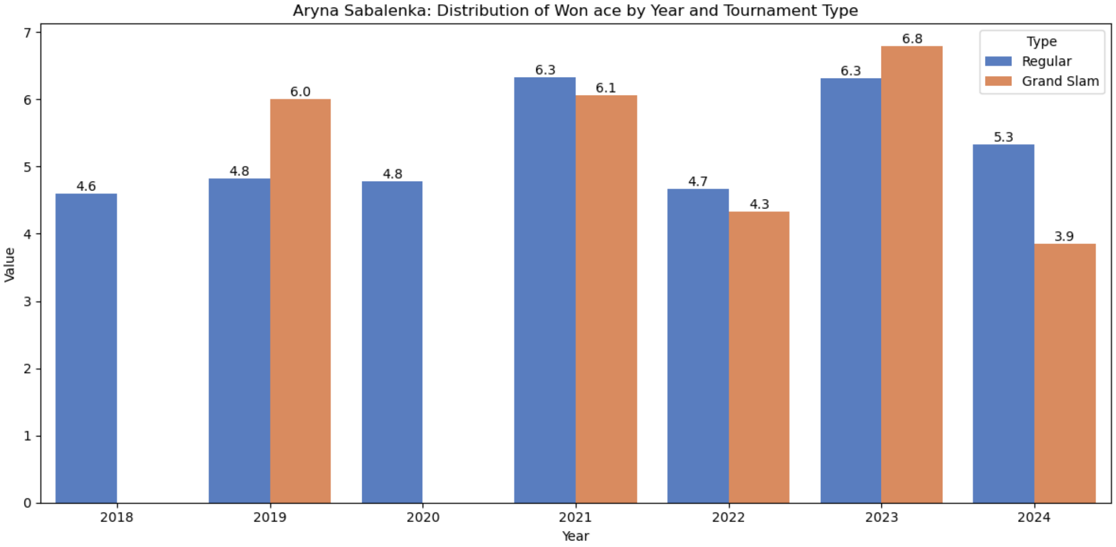


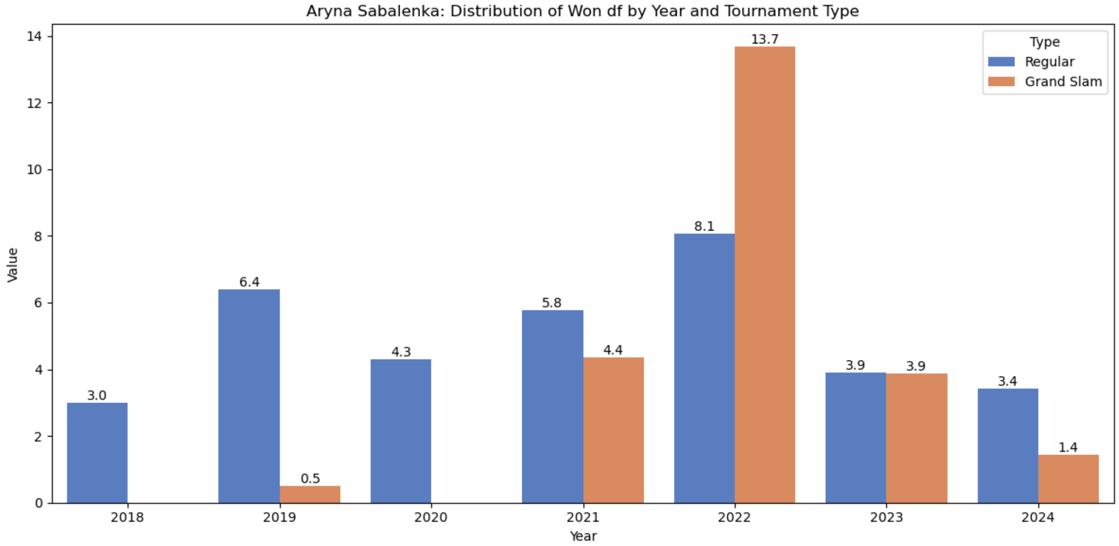
```
[40]: # Melting the DataFrame for easier plotting
      ASw_melted = ASw_summary.melt(id_vars=['Year', 'Type'], value_vars=['1stIn',␣
        ↪'1stWon', '2ndWon', 'ace', 'df'],
                          var_name='Metric', value_name='Value')

      for metric in metrics:
          plt.figure(figsize=(12, 6))
          ax = sns.barplot(data=ASw_melted[ASw_melted['Metric'] == metric], x='Year',␣
        ↪y='Value', hue='Type', errorbar=None, palette='muted')
          ax.set_title(f'Aryna Sabalenka: Distribution of Won {metric} by Year and␣
        ↪Tournament Type')
          ax.set_xlabel('Year')
          ax.set_ylabel('Value')
          ax.legend(title='Type')

          # Adding data labels
          for p in ax.patches:
              ax.annotate(f'{p.get_height():.1f}',
                          (p.get_x() + p.get_width() / 2., p.get_height()),
                          ha='center', va='bottom')

          plt.tight_layout()
          plt.show()
```


Aryna Sabalenka: Distribution of Won 1stIn by Year and Tournament Type
Type
Regular
Grand Slam
Value
Year
0
10
20
30
40
50
2018
2019
2020
2021
2022
2023
2024
42.2
45.5
36.5
45.3
40.7
36.7
45.6
52.7
38.8
37.9
51.0
32.1

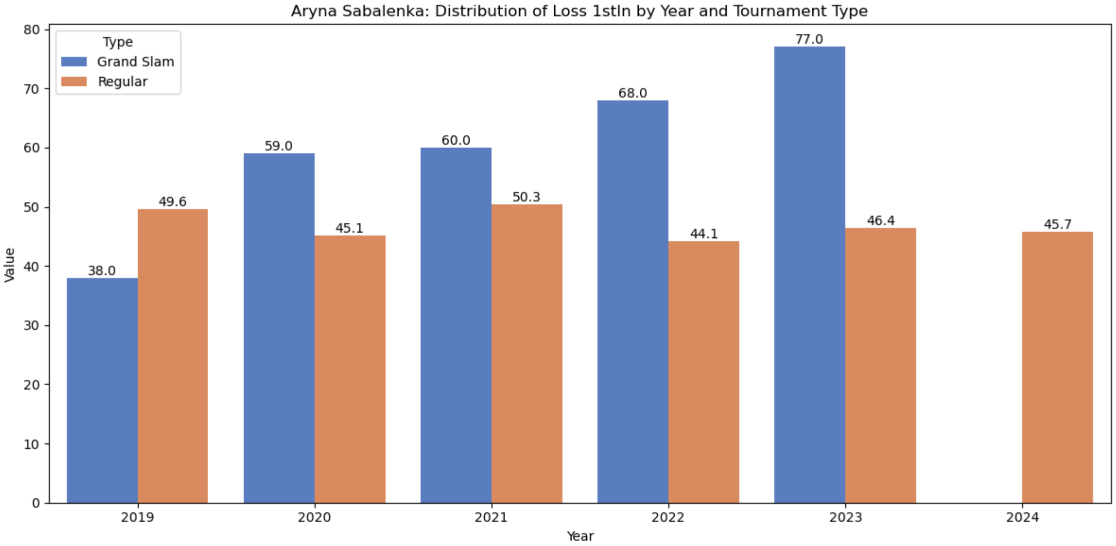

Aryna Sabalenka: Distribution of Won 1stWon by Year and Tournament Type
Type
Regular
Grand Slam
Value
Year
0
5
10
15
20
25
30
35
2018
2019
2020
2021
2022
2023
2024
28.6
31.5
30.0
31.7
29.9
25.6
31.7
36.7
29.1
28.5
37.2
25.1

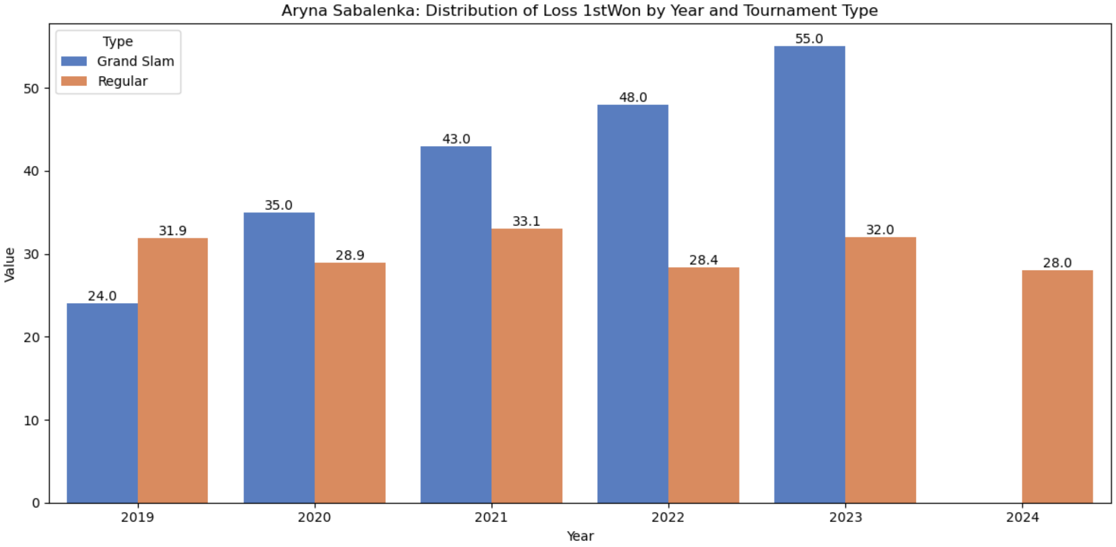

Aryna Sabalenka: Distribution of Won 2ndWon by Year and Tournament Type
Type
Regular
Grand Slam
Value
Year
0
2
4
6
8
10
12
14
2018
2019
2020
2021
2022
2023
2024
13.0
12.5
9.5
15.2
14.2
14.0
13.5
14.7
13.0
12.5
13.0
9.0

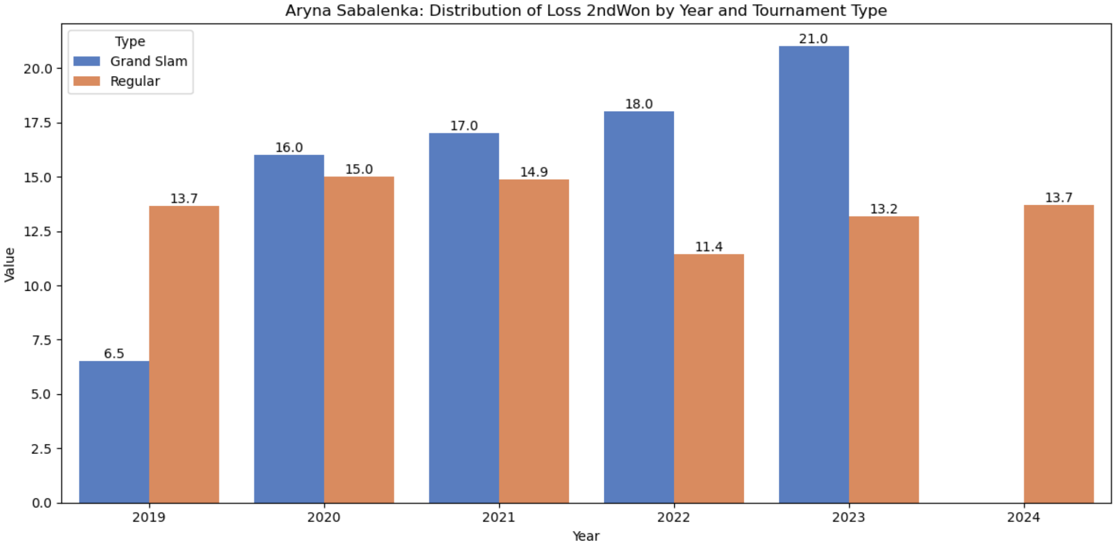

Aryna Sabalenka: Distribution of Won ace by Year and Tournament Type
Type
Regular
Grand Slam
Value
Year
0
1
2
3
4
5
6
7
2018
2019
2020
2021
2022
2023
2024
4.6
4.8
6.0
4.8
6.3
6.1
4.7
4.3
6.3
6.8
5.3
3.9

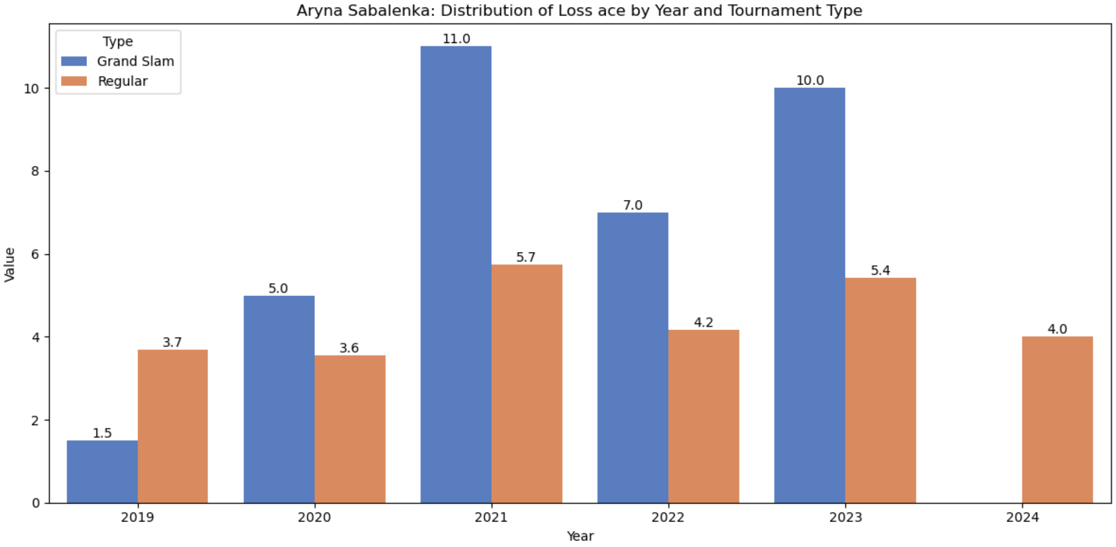


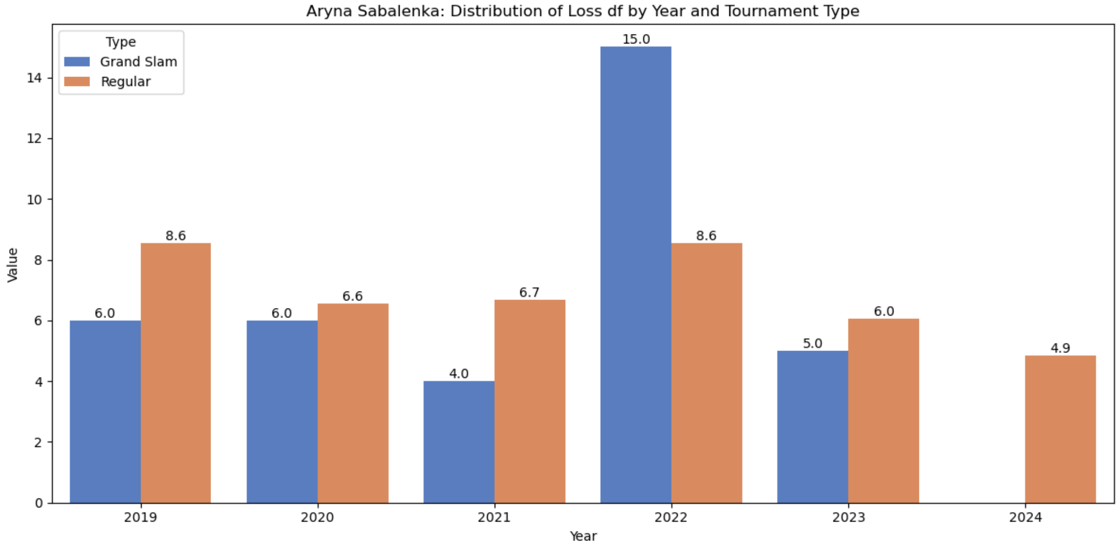
[41]:
```python
AS1_melted = AS1_summary.melt(id_vars=['Year', 'Type'], value_vars=['1stIn', '1stWon', '2ndWon', 'ace', 'df'],
                    var_name='Metric', value_name='Value')

for metric in metrics:
    plt.figure(figsize=(12, 6))
    ax = sns.barplot(data=AS1_melted[AS1_melted['Metric'] == metric], x='Year', y='Value', hue='Type', errorbar=None, palette='muted')
    ax.set_title(f'Aryna Sabalenka: Distribution of Loss {metric} by Year and Tournament Type')
    ax.set_xlabel('Year')
    ax.set_ylabel('Value')
    ax.legend(title='Type')

    # Adding data labels
    for p in ax.patches:
        ax.annotate(f'{p.get_height():.1f}',
                    (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom')

    plt.tight_layout()
    plt.show()
```


Aryna Sabalenka: Distribution of Loss 1stIn by Year and Tournament Type
Type
Grand Slam
Regular
Value
Year
0
10
20
30
40
50
60
70
80
2019
2020
2021
2022
2023
2024
38.0
49.6
59.0
45.1
60.0
50.3
68.0
44.1
77.0
46.4
45.7

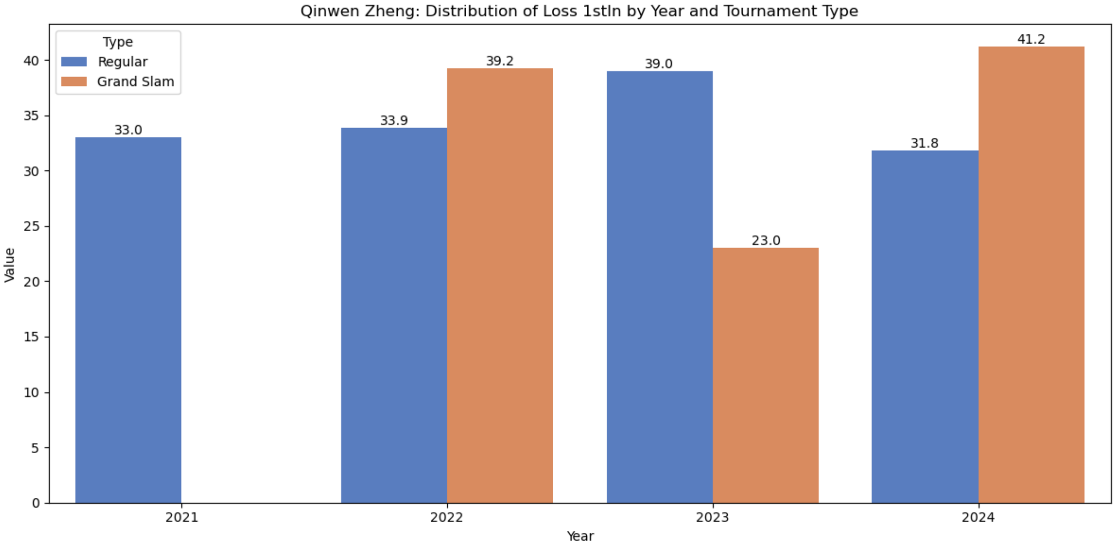

Aryna Sabalenka: Distribution of Loss 1stWon by Year and Tournament Type
Type
Grand Slam
Regular
Value
Year
0
10
20
30
40
50
2019
2020
2021
2022
2023
2024
24.0
31.9
35.0
28.9
43.0
33.1
48.0
28.4
55.0
32.0
28.0

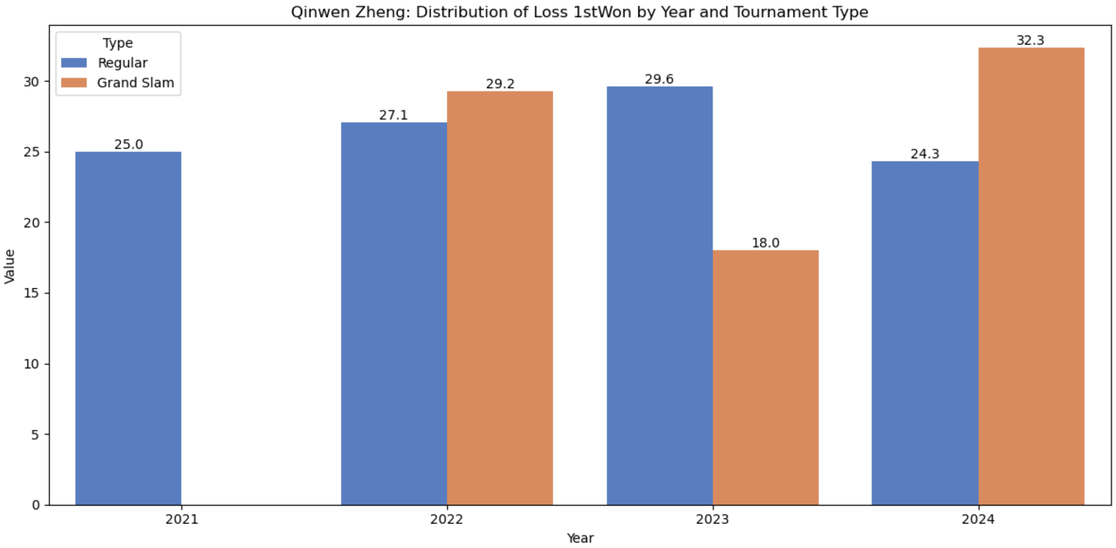

Aryna Sabalenka: Distribution of Loss 2ndWon by Year and Tournament Type
Type
Grand Slam
Regular
Value
Year
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
2019
2020
2021
2022
2023
2024
6.5
13.7
16.0
15.0
17.0
14.9
18.0
11.4
21.0
13.2
13.7

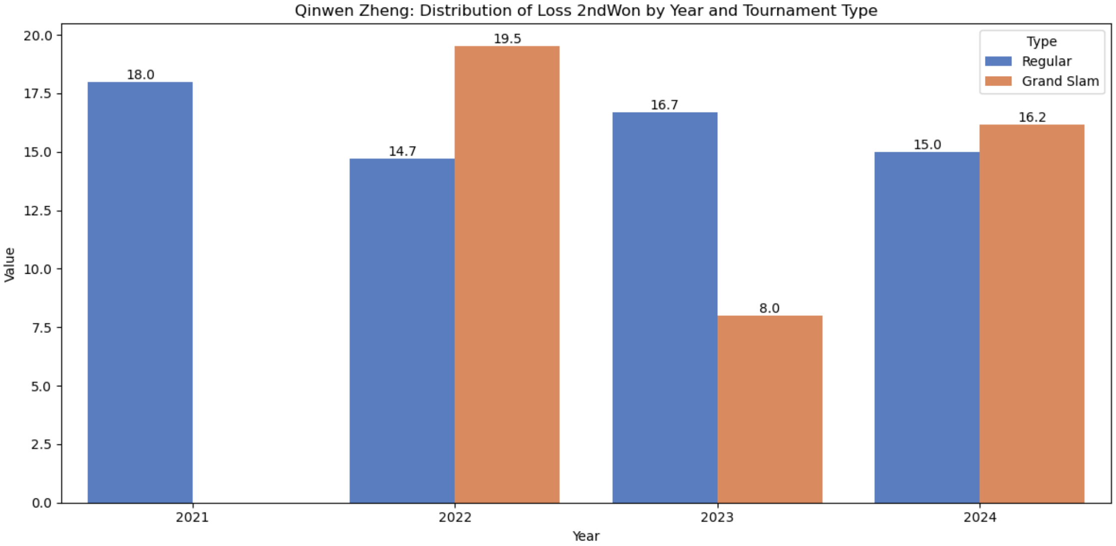

Aryna Sabalenka: Distribution of Loss ace by Year and Tournament Type
Type
Grand Slam
Regular
Value
Year
0
2
4
6
8
10
2019
2020
2021
2022
2023
2024
1.5
3.7
5.0
3.6
11.0
5.7
7.0
4.2
10.0
5.4
4.0

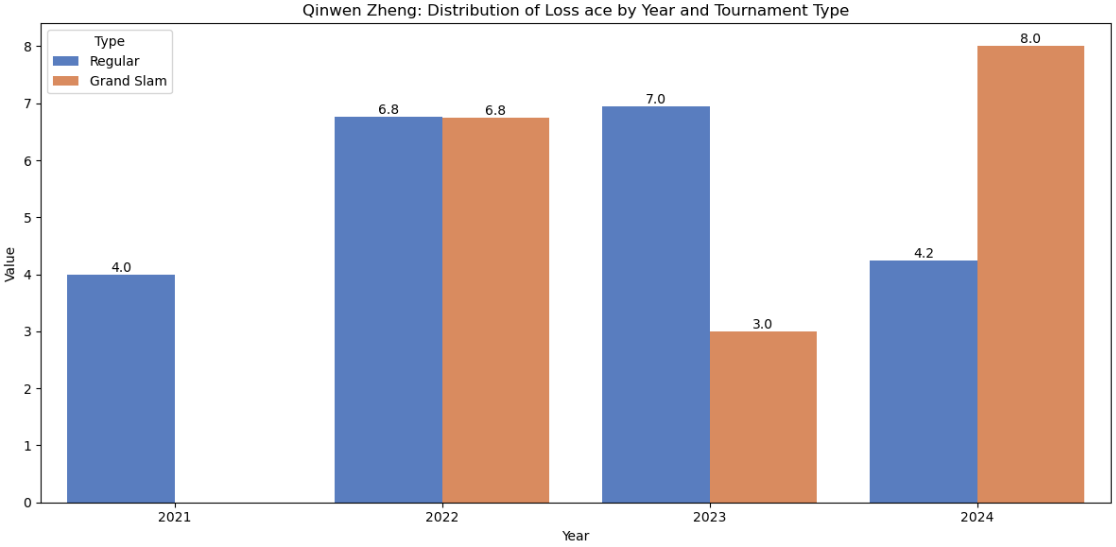


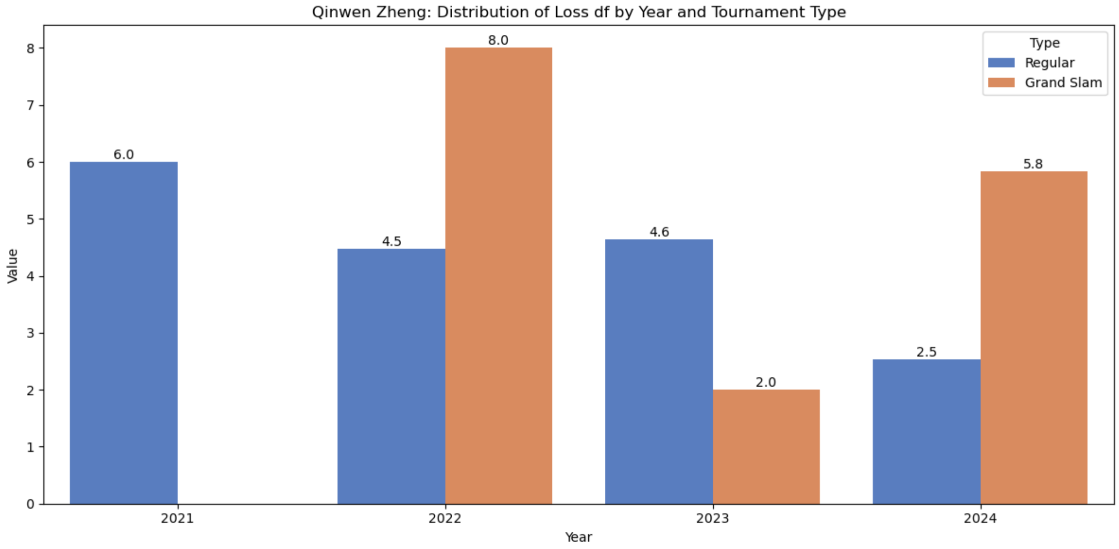
```
[154]: QZw_melted = QZw_summary.melt(id_vars=['Year', 'Type'], value_vars=['1stIn',␣
        ↪'1stWon', '2ndWon', 'ace', 'df'],
                            var_name='Metric', value_name='Value')

       for metric in metrics:
           plt.figure(figsize=(12, 6))
           ax = sns.barplot(data=QZw_melted[QZw_melted['Metric'] == metric], x='Year',␣
        ↪y='Value', hue='Type', errorbar=None, palette='muted')
           ax.set_title(f'Qinwen Zheng: Distribution of Loss {metric} by Year and␣
        ↪Tournament Type')
           ax.set_xlabel('Year')
           ax.set_ylabel('Value')
           ax.legend(title='Type')

           # Adding data labels
           for p in ax.patches:
               ax.annotate(f'{p.get_height():.1f}',
                           (p.get_x() + p.get_width() / 2., p.get_height()),
                           ha='center', va='bottom')

           plt.tight_layout()
           plt.show()
```


Qinwen Zheng: Distribution of Loss 1stIn by Year and Tournament Type
Type
Regular
Grand Slam
Value
Year
0
5
10
15
20
25
30
35
40
2021
2022
2023
2024
33.0
33.9
39.2
39.0
23.0
31.8
41.2

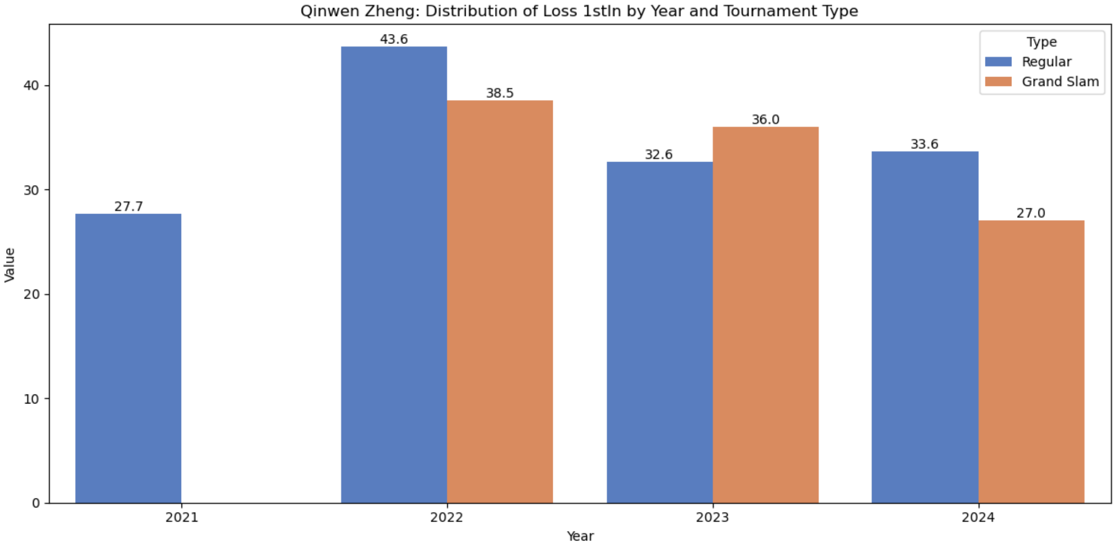

Qinwen Zheng: Distribution of Loss 1stWon by Year and Tournament Type
Type
Regular
Grand Slam
Value
Year
0
5
10
15
20
25
30
2021
2022
2023
2024
25.0
27.1
29.2
29.6
18.0
24.3
32.3

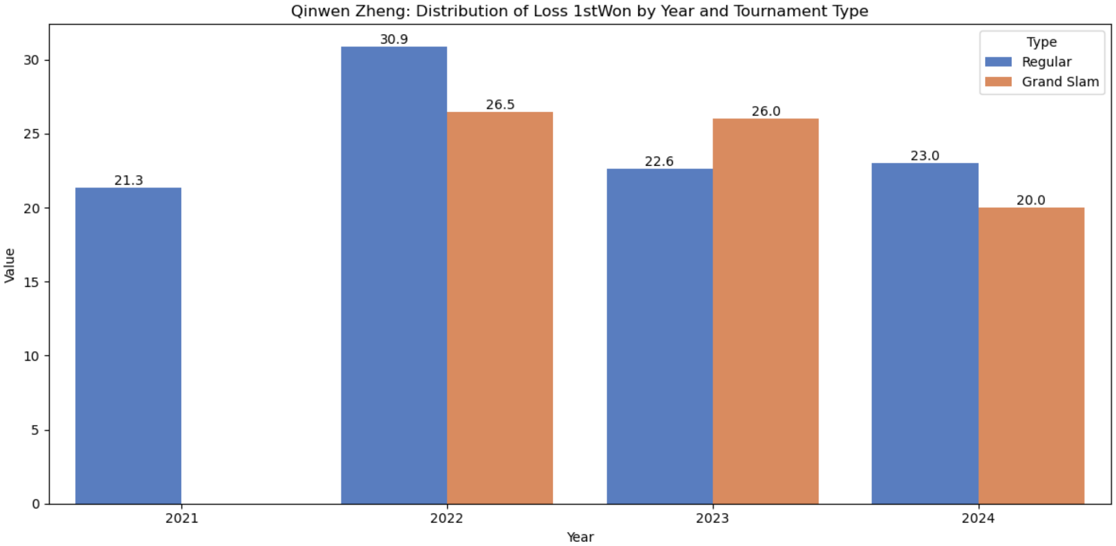

Qinwen Zheng: Distribution of Loss 2ndWon by Year and Tournament Type
Type
Regular
Grand Slam
Value
Year
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
2021
2022
2023
2024
18.0
14.7
19.5
16.7
8.0
15.0
16.2

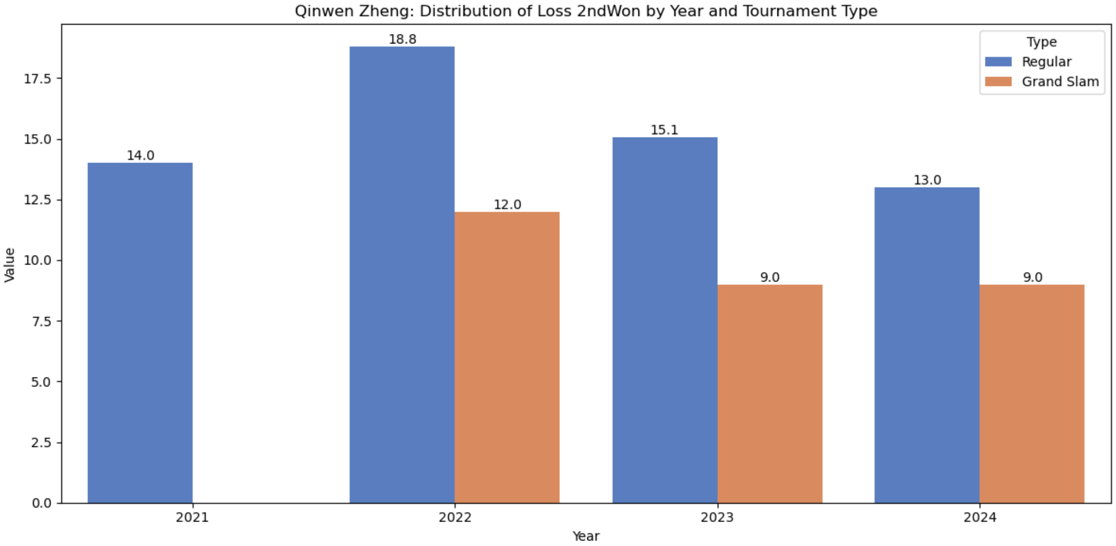

Qinwen Zheng: Distribution of Loss ace by Year and Tournament Type
Type
Regular
Grand Slam
Value
Year
8
7
6
5
4
3
2
1
0
2021
2022
2023
2024
4.0
6.8
6.8
7.0
3.0
4.2
8.0

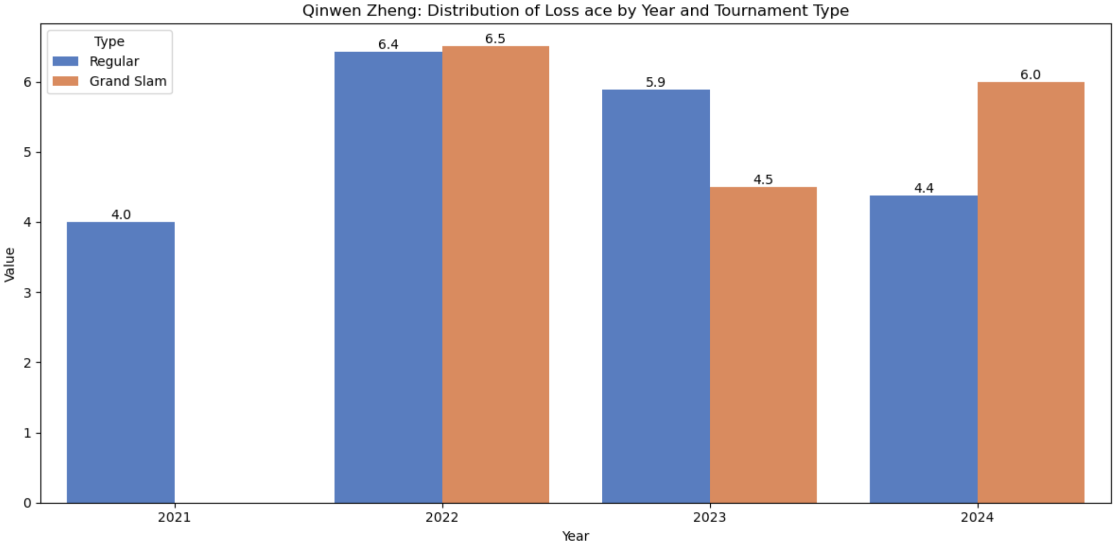


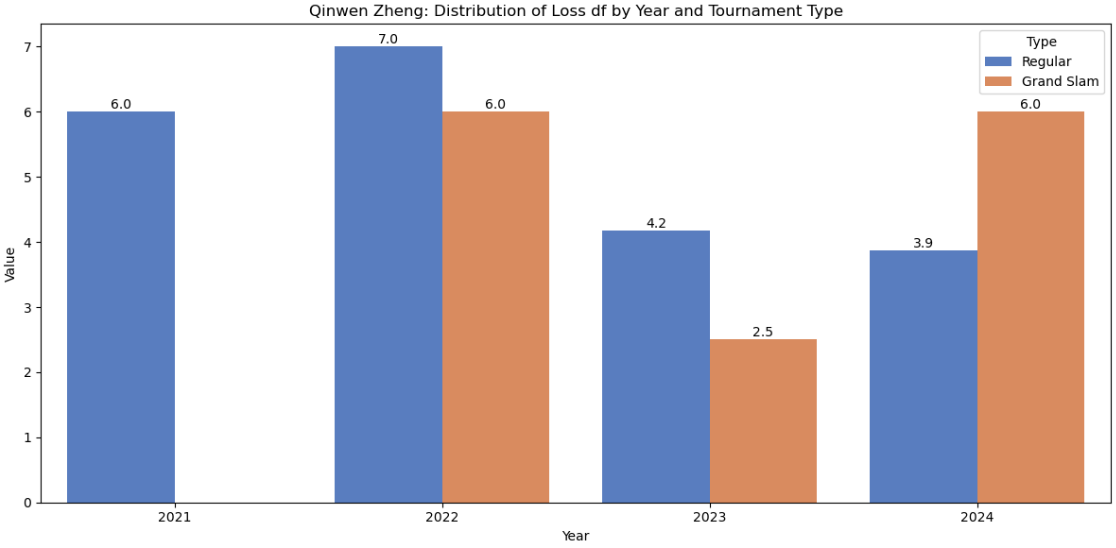
[155]:
```
QZl_melted = QZl_summary.melt(id_vars=['Year', 'Type'], value_vars=['1stIn',␣
 ↪'1stWon', '2ndWon', 'ace', 'df'],
                    var_name='Metric', value_name='Value')

for metric in metrics:
    plt.figure(figsize=(12, 6))
    ax = sns.barplot(data=QZl_melted[QZl_melted['Metric'] == metric], x='Year',␣
 ↪y='Value', hue='Type', errorbar=None, palette='muted')
    ax.set_title(f'Qinwen Zheng: Distribution of Loss {metric} by Year and␣
 ↪Tournament Type')
    ax.set_xlabel('Year')
    ax.set_ylabel('Value')
    ax.legend(title='Type')

    # Adding data labels
    for p in ax.patches:
        ax.annotate(f'{p.get_height():.1f}',
                    (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom')

    plt.tight_layout()
    plt.show()
```


Qinwen Zheng: Distribution of Loss 1stIn by Year and Tournament Type
Type
Regular
Grand Slam
Value
0
10
20
30
40
Year
2021
2022
2023
2024
27.7
43.6
38.5
32.6
36.0
33.6
27.0


Qinwen Zheng: Distribution of Loss 1stWon by Year and Tournament Type
Type
Regular
Grand Slam
Value
0
5
10
15
20
25
30
Year
2021
2022
2023
2024
21.3
30.9
26.5
22.6
26.0
23.0
20.0


Qinwen Zheng: Distribution of Loss 2ndWon by Year and Tournament Type
Type
Regular
Grand Slam
Value
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
14.0
18.8
12.0
15.1
9.0
13.0
9.0
2021
2022
2023
2024
Year


Qinwen Zheng: Distribution of Loss ace by Year and Tournament Type
Type
Regular
Grand Slam
Value
0
1
2
3
4
5
6
4.0
6.4
6.5
5.9
4.5
4.4
6.0
2021
2022
2023
2024
Year


Qinwen Zheng: Distribution of Loss df by Year and Tournament Type
Type
Regular
Grand Slam
Value
Year
0
1
2
3
4
5
6
7
2021
2022
2023
2024
6.0
7.0
6.0
4.2
2.5
3.9
6.0

```
[ ]:
```

CocoGauff

November 13, 2024

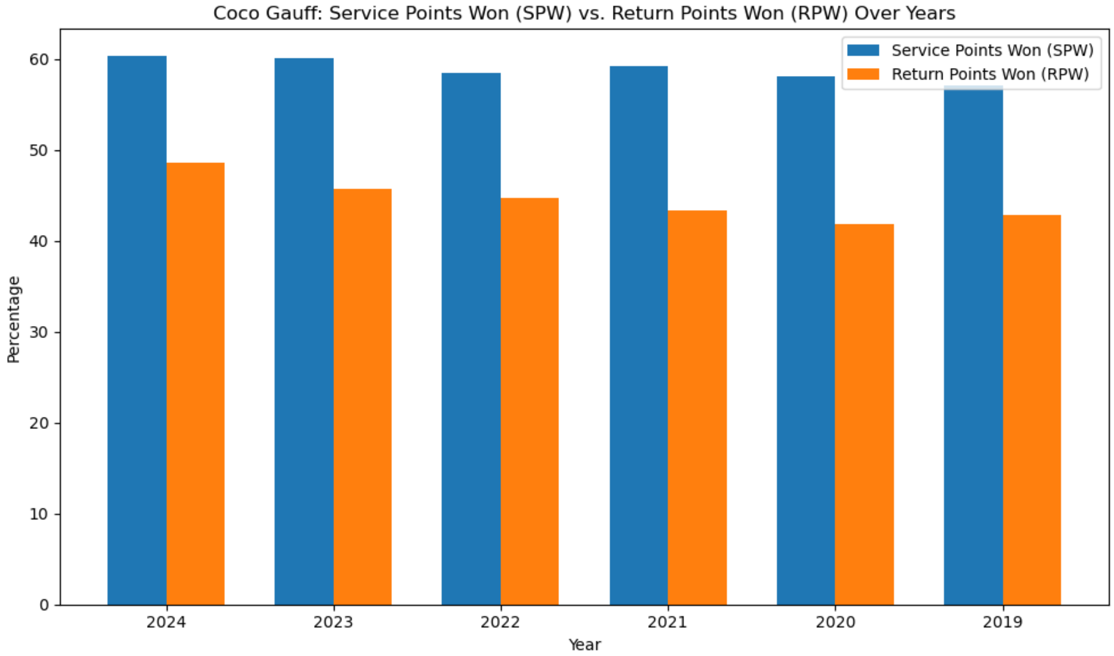
[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[4]:
```
file_path = "C:\\Users\\maldo\\Downloads\\CocoG.csv"
coco_data = pd.read_csv(file_path)

# Display the first few rows of the dataset to understand its columns and data␣
 ↪types
coco_data.head()
```

[4]:
```
   Year   M   W   L   Win% Set W-L   Set% Game W-L   Game% TB W-L  …    A%  \
0  2024  66  50  16  75.8%  106-44  70.7%  808-539  60.0%     8-7  …  5.2%
1  2023  67  51  16  76.1%  107-43  71.3%  776-563  58.0%     8-1  …  6.0%
2  2022  61  38  23  62.3%   80-51  61.1%  672-572  54.0%    10-7  …  6.0%
3  2021  50  34  16  68.0%   69-46  60.0%  590-514  53.4%    10-7  …  6.8%
4  2020  18  10   8  55.6%   26-19  57.8%  229-217  51.3%     3-2  …  4.0%

     DF%  1stIn   1st%   2nd%    SPW    RPW    TPW    DR               Best
0   9.2%  57.5%  72.2%  44.1%  60.3%  48.6%  54.4%  1.22             W (2x)
1   5.1%  60.7%  68.1%  47.6%  60.1%  45.7%  52.9%  1.14             W (4x)
2   7.5%  63.0%  65.9%  45.8%  58.5%  44.7%  51.5%  1.08  F (Roland Garros)
3   7.2%  60.4%  67.7%  46.2%  59.2%  43.4%  51.3%  1.06          W (Parma)
4  10.6%  61.0%  68.8%  41.3%  58.1%  41.9%  50.2%  1.00    SF (Lexington)

[5 rows x 24 columns]
```

[5]:
```
# Convert SPW and RPW columns to numerical values (remove percentage signs and␣
 ↪convert to float)
coco_data['SPW'] = coco_data['SPW'].str.replace('%', '').astype(float)
coco_data['RPW'] = coco_data['RPW'].str.replace('%', '').astype(float)

# Set up the grouped bar chart data
years = coco_data['Year']
spw_values = coco_data['SPW']
rpw_values = coco_data['RPW']
```

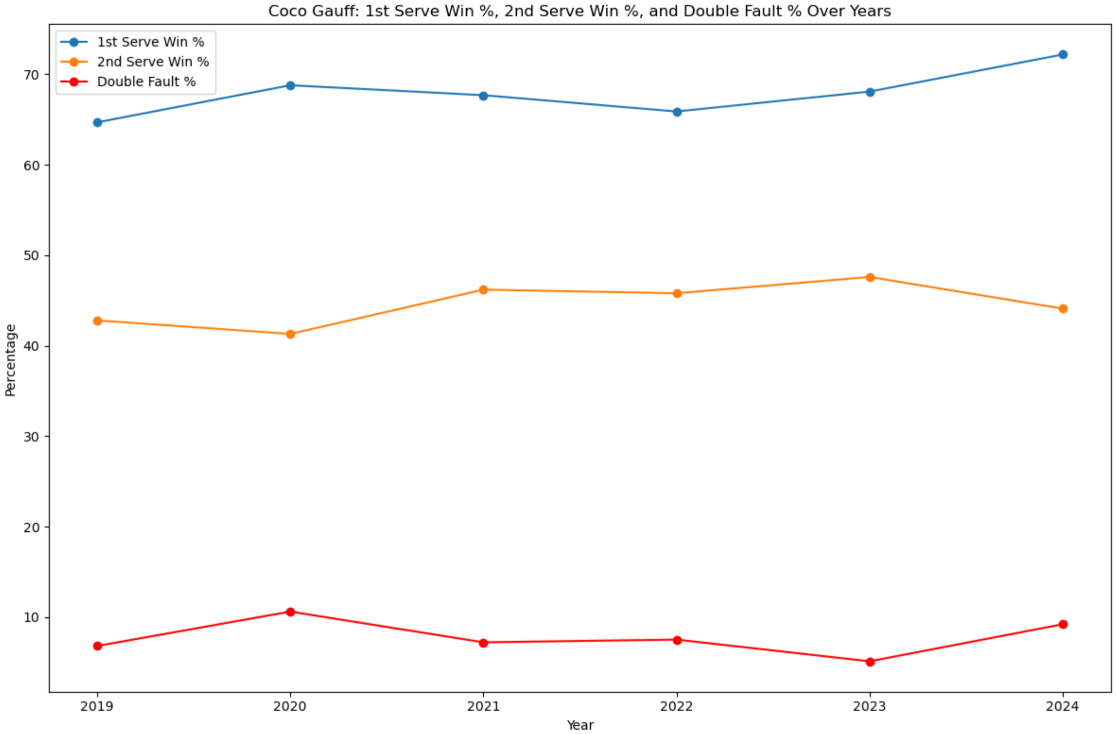
```
# Plotting
plt.figure(figsize=(10, 6))
bar_width = 0.35
index = range(len(years))

# Bars for Service Points Won (SPW) and Return Points Won (RPW)
plt.bar(index, spw_values, width=bar_width, label='Service Points Won (SPW)')
plt.bar([i + bar_width for i in index], rpw_values, width=bar_width,␣
  ↪label='Return Points Won (RPW)')

# Labels and Title
plt.xlabel('Year')
plt.ylabel('Percentage')
plt.title('Coco Gauff: Service Points Won (SPW) vs. Return Points Won (RPW)␣
  ↪Over Years')
plt.xticks([i + bar_width / 2 for i in index], years)
plt.legend()

plt.tight_layout()
plt.show()
```



[6]:

```
# Convert 1st%, 2nd%, and DF% columns to numerical values (remove percentage␣
  ↪signs and convert to float)
coco_data['1st%'] = coco_data['1st%'].str.replace('%', '').astype(float)
coco_data['2nd%'] = coco_data['2nd%'].str.replace('%', '').astype(float)
```

```
coco_data['DF%'] = coco_data['DF%'].str.replace('%', '').astype(float)

# Plotting 1st Serve Win Percentage, 2nd Serve Win Percentage, and Double Fault Percentage over the years
plt.figure(figsize=(12, 8))

# 1st Serve Win Percentage
plt.plot(coco_data['Year'], coco_data['1st%'], marker='o', linestyle='-', label='1st Serve Win %')

# 2nd Serve Win Percentage
plt.plot(coco_data['Year'], coco_data['2nd%'], marker='o', linestyle='-', label='2nd Serve Win %')

# Double Fault Percentage
plt.plot(coco_data['Year'], coco_data['DF%'], marker='o', linestyle='-', color='r', label='Double Fault %')

# Labels and Title
plt.xlabel('Year')
plt.ylabel('Percentage')
plt.title('Coco Gauff: 1st Serve Win %, 2nd Serve Win %, and Double Fault % Over Years')
plt.legend()

plt.tight_layout()
plt.show()
```

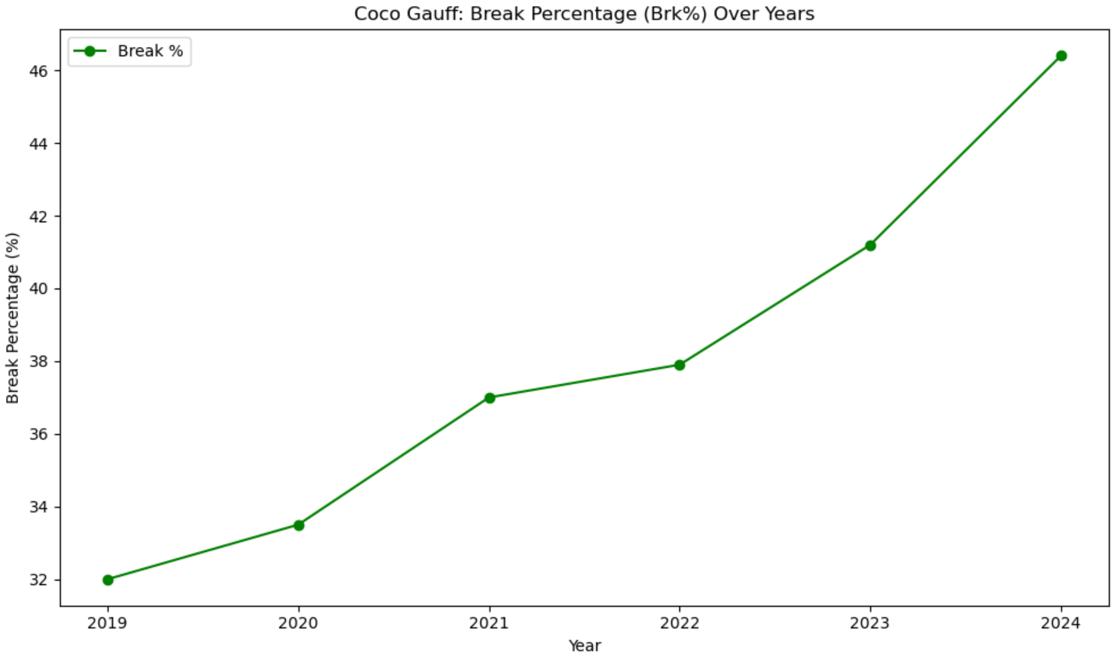


The line graph shows a higher 1st serve win % than 2nd serve win % and double faults, indicating consistency in this area and a reliance on her first serve. The inflection points on the 2nd serve win % line seem to be inversely related to the inflection points on the double fault % line, indicating that when her 2nd serve win % is low, her double fault % increases and vice versa. This reflects a strategic balance. Higher Double Fault Percentage usually suggests that she's taking more risks on her second serve, possibly going for more aggressive serves to avoid putting her opponent in a strong return position. This could result in fewer points won on second serves if the risk doesn't pay off. Higher 2nd Serve Win Percentage alongside a lower Double Fault Percentage implies that she's likely playing it safer, focusing on keeping the ball in play rather than going for high-risk serves. So, when her 2nd serve win percentage is low, her double fault percentage tends to increase, which aligns with the idea of a riskier second serve approach. Conversely, when she's winning more on her second serve, it's likely due to more conservative second serves, thus reducing double faults. This analysis points to her reliance on the first serve as her primary strategy and her adjustments on the second serve based on the need for balance between risk and consistency.

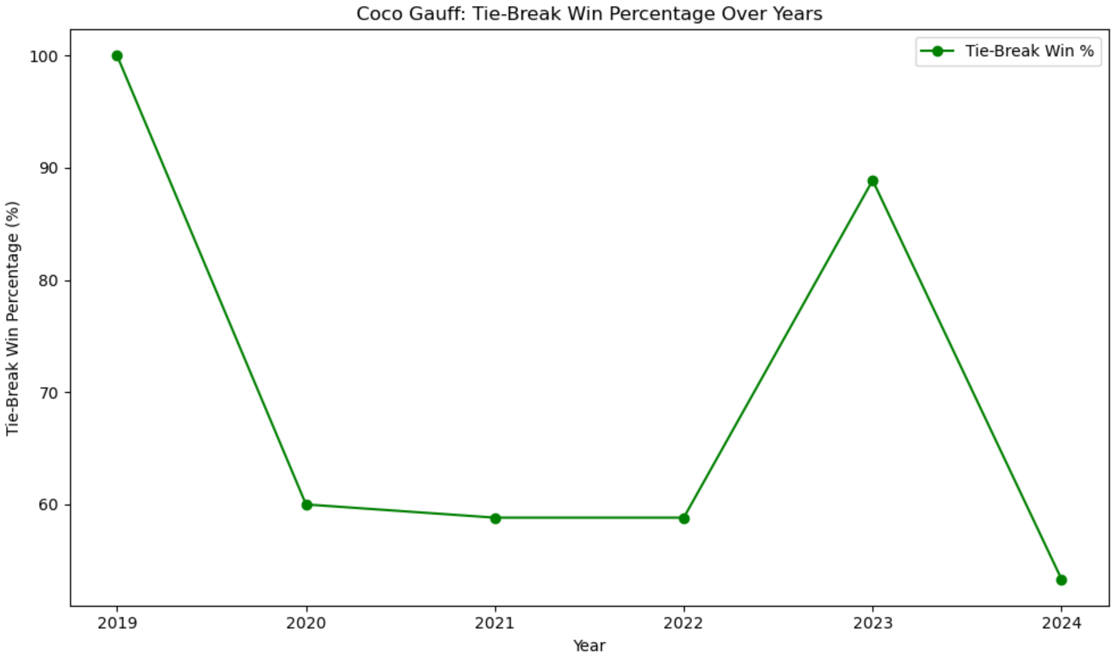
[7]:

```
# Convert Brk% column to a numerical value (remove percentage signs and convert␣
  ↪to float)
coco_data['Brk%'] = coco_data['Brk%'].str.replace('%', '').astype(float)

# Plotting Break Percentage (Brk%) over the years
plt.figure(figsize=(10, 6))
plt.plot(coco_data['Year'], coco_data['Brk%'], marker='o', linestyle='-',␣
  ↪color='g', label='Break %')
```

```
# Labels and Title
plt.xlabel('Year')
plt.ylabel('Break Percentage (%)')
plt.title('Coco Gauff: Break Percentage (Brk%) Over Years')
plt.legend()

plt.tight_layout()
plt.show()
```



[9]:
```
# Split the TB W-L column into separate "Win" and "Loss" columns
coco_data[['TB Wins', 'TB Losses']] = coco_data['TB W-L'].str.split('-',␣
 ↪expand=True).astype(int)

# Calculate the Tie-Break Win Percentage
coco_data['TB Win%'] = coco_data['TB Wins'] / (coco_data['TB Wins'] +␣
 ↪coco_data['TB Losses']) * 100

# Plotting Tie-Break Win Percentage (TB Win%) over the years
plt.figure(figsize=(10, 6))
plt.plot(coco_data['Year'], coco_data['TB Win%'], marker='o', linestyle='-',␣
 ↪color='g', label='Tie-Break Win %')

# Labels and Title
plt.xlabel('Year')
plt.ylabel('Tie-Break Win Percentage (%)')
plt.title('Coco Gauff: Tie-Break Win Percentage Over Years')
plt.legend()

plt.tight_layout()
plt.show()
```



[ ]: