

```
import pandas as pd
import requests

# Wikipedia page (example)
url = "https://en.wikipedia.org/wiki/Bradby_Shield_Encounter"

# Send request with headers to avoid 403
headers = {"User-Agent": "Mozilla/5.0"}
response = requests.get(url, headers=headers)

# Read all tables from the HTML response
tables = pd.read_html(response.text)

print(f"Total tables found: {len(tables)}")

/ tmp/ipython-input-2278338758.py:9: FutureWarning: Passing literal html to 'read_html' is deprecated and will be removed
  tables = pd.read_html(response.text)
Total tables found: 28

# Pick one table (try different indices to see which is the right one)
df = tables[13] # adjust index if needed

# Preview
df
```

Bradby Shield Encounter: 1945–present

	Series	Year	Captains		Date	Venue	Score		Winner	Ref.
	Series	Year	Royal	Trinity	Date	Venue	Royal	Trinity	Winner	Ref.
0	1	1945	C. D. L. Fernando	R. G. Sourjah (1L) S. B. Pilapitiya (2L)	13 July	Racecourse	3	0	Trinity	NaN
1	1	1945	C. D. L. Fernando	R. G. Sourjah (1L) S. B. Pilapitiya (2L)	20 July	Bogambara	0	6	Trinity	NaN
2	2	1946	M. Rodrigo	S. B. Pilapitiya	6 July	Police Park	3	0	Trinity (2)	NaN
3	2	1946	M. Rodrigo	S. B. Pilapitiya	20 July	Bogambara	0	8	Trinity (2)	NaN
4	3	1947	N. W. Karunaratne	G. Sanmugam	12 July	Bogambara	0	6	Trinity (3)	NaN
...
157	77	2023	R. Senanayake	A. Manzil	5 August	RCSC	27	17	Royal (36)	NaN
158	78	2024	T. Perera	T. Dissanayaka[d]	24 August	RCSC	16	17	Trinity (40)	[19][20]
159	78	2024	T. Perera	T. Dissanayaka[d]	7 September	TCRS	23	25	Trinity (40)	[19][20]

Next steps: [Generate code with df](#) [View recommended plots](#) [New interactive sheet](#)

```
from google.colab import files

# Upload CSV file from your computer
uploaded = files.upload() # This will open a file picker

Choose files bradby_df.csv
• bradby_df.csv(text/csv) - 12866 bytes, last modified: 31/08/2025 - 100% done
Saving bradby_df.csv to bradby_df.csv

# Get the uploaded file name
filename = list(uploaded.keys())[0]

# Load the CSV into a DataFrame
df = pd.read_csv(filename)

df
```

	Series	Year	Royal Skipper	Trinity Skipper	Date	Venue	Royal	Trinity	1st Leg Winer	2nd Leg Winner	Bradby Winner
0	1	1945	C. D. L. Fernando	R. G. Sourjah (1L) S. B. Pilapitiya (2L)	13 July	Racecourse	3	0	NaN	NaN	Trinity
1	1	1945	C. D. L. Fernando	R. G. Sourjah (1L) S. B. Pilapitiya (2L)	20 July	Bogambara	0	6	NaN	NaN	Trinity
2	2	1946	M. Rodrigo	S. B. Pilapitiya	6 July	Police Park	3	0	NaN	NaN	Trinity (2)
3	2	1946	M. Rodrigo	S. B. Pilapitiya	20 July	Bogambara	0	8	NaN	NaN	Trinity (2)
4	3	1947	N. W. Karunaratne	G. Sanmugam	12 July	Bogambara	0	6	NaN	NaN	Trinity (3)
...
157	77	2023	R. Senanayake	A. Manzil	5 August	RCSC	27	17	NaN	NaN	Royal (36)
158	78	2024	T. Perera	T. Dissanayaka[d]	24 August	RCSC	16	17	NaN	NaN	Trinity (40)
159	78	2024	T. Perera	T. Dissanayaka[d]	7 September	TCRS	23	25	NaN	NaN	Trinity (40)

Next steps: [Generate code with df](#) [View recommended plots](#) [New interactive sheet](#)

```
#getting unique values for venues
print(df['Venue'].unique())

['Racecourse' 'Bogambara' 'Police Park' 'Reid Avenue' 'Longden Place'
 'Nittawela' 'Havelock Park' 'Peradeniya'
 'Second leg not played due to 1971 JVP insurrection' 'Maitland Crescent'
 'Sugathadasa' 'TCRS' 'RCSC' '-']

# Changing different venue names to city for simplicity
colombo_grounds = ['Racecourse', 'Police Park', 'Reid Avenue', 'Longden Place',
                   'Havelock Park', 'Sugathadasa', 'RCSC', 'Maitland Crescent']
kandy_grounds = ['Nittawela', 'Peradeniya', 'TCRS', 'Bogambara']

# Replace values in the column
df['Venue'] = df['Venue'].replace(colombo_grounds, 'Colombo')
df['Venue'] = df['Venue'].replace(kandy_grounds, 'Kandy')

# Preview
print(df['Venue'].unique())

['Colombo' 'Kandy' 'Second leg not played due to 1971 JVP insurrection'
 '-']
```

df

	Series	Year	Royal Skipper	Trinity Skipper	Date	Venue	Royal	Trinity	1st Leg Winer	2nd Leg Winner	Bradby Winner
0	1	1945	C. D. L. Fernando	R. G. Sourjah (1L) S. B. Pilapitiya (2L)	13 July	Colombo	3	0	NaN	NaN	Trinity
1	1	1945	C. D. L. Fernando	R. G. Sourjah (1L) S. B. Pilapitiya (2L)	20 July	Kandy	0	6	NaN	NaN	Trinity
2	2	1946	M. Rodrigo	S. B. Pilapitiya	6 July	Colombo	3	0	NaN	NaN	Trinity (2)
3	2	1946	M. Rodrigo	S. B. Pilapitiya	20 July	Kandy	0	8	NaN	NaN	Trinity (2)
4	3	1947	N. W. Karunaratne	G. Sanmugam	12 July	Kandy	0	6	NaN	NaN	Trinity (3)
...
157	77	2023	R. Senanayake	A. Manzil	5 August	Colombo	27	17	NaN	NaN	Royal (36)
158	78	2024	T. Perera	T. Dissanayaka[d]	24 August	Colombo	16	17	NaN	NaN	Trinity (40)
159	78	2024	T. Perera	T. Dissanayaka[d]	7 September	Kandy	23	25	NaN	NaN	Trinity (40)

Next steps: [Generate code with df](#) [View recommended plots](#) [New interactive sheet](#)

```
# List of unwanted values
unwanted_values = ['Second leg not played due to 1971 JVP insurrection', '-']

# Remove rows where 'Venue' column has any of these values
df = df[~df['Venue'].isin(unwanted_values)].reset_index(drop=True)
```

```

import numpy as np

# Define conditions and choices
conditions = [
    df['Bradby Winner'].str.contains('Trinity', case=False, na=False),
    df['Bradby Winner'].str.contains('Royal', case=False, na=False)
]

choices = ['Trinity', 'Royal']

# Create new column 'Winner'
df['Winner'] = np.select(conditions, choices, default=df['Bradby Winner'])

# Rename 'Bradby Winner.' column to 'Cumulative'
df = df.rename(columns={'Bradby Winner': 'Cumulative'})

df = df.rename(columns={
    '1st Leg Winer': 'Royal 1st leg',
    '2nd Leg Winner': 'Royal 2nd leg'
})

df['Trinity 1st leg'] = ""
df['Trinity 2nd leg'] = ""

df['1st Leg Venue'] = ""
df['2nd Leg Venue'] = ""

for year in df['Year'].unique():
    # Get indices of rows for this year
    year_indices = df[df['Year'] == year].index

    # Assign Venue values to 1st and 2nd leg columns
    if len(year_indices) >= 2:
        first_idx = year_indices[0]
        second_idx = year_indices[1]

        df.at[first_idx, '1st leg Venue'] = df.at[first_idx, 'Venue']
        df.at[second_idx, '2nd leg Venue'] = df.at[second_idx, 'Venue']

import numpy as np

# Replace NaN in '2nd leg venue' based on '1st leg venue'
df['2nd leg Venue'] = np.where(
    df['2nd leg Venue'].isna() & (df['1st leg Venue'] == 'Colombo'), 'Kandy',
    np.where(
        df['2nd leg Venue'].isna() & (df['1st leg Venue'] == 'Kandy'), 'Colombo',
        df['2nd leg Venue'] # keep existing value if not NaN
    )
)

df = df.iloc[:-1].reset_index(drop=True)

# Convert the 1st/2nd leg columns to numeric (float) first
df['Royal 1st leg'] = df['Royal 1st leg']
df['Royal 2nd leg'] = df['Royal 2nd leg']

# Then assign values
for year in df['Year'].unique():
    year_indices = df[df['Year'] == year].index

    if len(year_indices) >= 2:
        first_idx = year_indices[0]
        second_idx = year_indices[1]

        df.at[first_idx, 'Royal 1st leg'] = df.at[first_idx, 'Royal']
        df.at[second_idx, 'Royal 2nd leg'] = df.at[second_idx, 'Royal']

⚠ /tmp/ipython-input-248567876.py:13: FutureWarning: Setting an item of incompatible dtype is deprecated and will raise an
df.at[first_idx, 'Royal 1st leg'] = df.at[first_idx, 'Royal']
/tmp/ipython-input-248567876.py:14: FutureWarning: Setting an item of incompatible dtype is deprecated and will raise an
df.at[first_idx, 'Royal 2nd leg'] = df.at[second_idx, 'Royal']

# Convert the 1st/2nd leg columns to numeric
df['Trinity 1st leg'] = df['Trinity 1st leg']

```

```
df['Trinity 2nd leg'] = df['Trinity 2nd leg']

# Then assign values
for year in df['Year'].unique():
    year_indices = df[df['Year'] == year].index

    if len(year_indices) >= 2:
        first_idx = year_indices[0]
        second_idx = year_indices[1]

        df.at[first_idx, 'Trinity 1st leg'] = df.at[first_idx, 'Trinity']
        df.at[first_idx, 'Trinity 2nd leg'] = df.at[second_idx, 'Trinity']

df = df.drop_duplicates(subset='Year', keep='first').reset_index(drop=True)
```

df



	Series	Year	Royal Skipper	Trinity Skipper	Date	Venue	Royal	Trinity	Royal 1st leg	Royal 2nd leg	Cumulative	Winner	Trinity 1st leg	Trinity 2nd leg
0	1	1945	C. D. L. Fernando	R. G. Sourjah (1L) S. B. Pilapitiya (2L)	13 July	Colombo	3	0	3	0	Trinity	Trinity	0	
1	2	1946	M. Rodrigo	S. B. Pilapitiya	6 July	Colombo	3	0	3	0	Trinity (2)	Trinity	0	
2	3	1947	N. W. Karunaratne	G. Sanmugam	12 July	Kandy	0	6	0	0	Trinity (3)	Trinity	6	
3	4	1948	A. A. Cader	A. R. A. Mohamed	3 July	Colombo	6	3	6	8	Royal	Royal	3	
4	5	1949	G. C. Weinman	K. Arumugam	3 July	Colombo	5	3	5	8	Royal (2)	Royal	3	
...
74	75	2019	T. Hassen	R. Bandaranayake	1 June	Kandy	34	17	34	13	Royal (34)	Royal	17	
75	76	2022	H. Peiris	L. Moragoda	20 August	Colombo	29	0	29	21	Royal (35)	Royal	0	
76	77	2023	R. Senanayake	A. Manzil	22 July	Kandy	10	13	10	27	Royal (36)	Royal	13	
77	78	2024	T. Perera	T. Dissanayaka[d]	24 August	Colombo	16	17	16	23	Trinity (40)	Trinity	17	
78	79	2025	A. Samarasinghe	M. U. Shafraz	23 August	Kandy	15	5	NaN	NaN	TBD (TBD)	TBD (TBD)		

79 rows x 18 columns

Next steps: [Generate code with df](#) [View recommended plots](#) [New interactive sheet](#)

```
columns_to_drop = ['Date', 'Venue', 'Royal', 'Trinity', '1st Leg Venue', '2nd Leg Venue']
df = df.drop(columns=columns_to_drop)

df = df.iloc[:-1].reset_index(drop=True)

df
```

	Series	Year	Royal Skipper	Trinity Skipper	Royal 1st leg	Royal 2nd leg	Cumulative	Winner	Trinity 1st leg	Trinity 2nd leg	1st leg Venue	2nd leg Venue
0	1	1945	C. D. L. Fernando	R. G. Sourjah (1L) S. B. Pilapitiya (2L)	3	0	Trinity	Trinity	0	6	Colombo	Kandy
1	2	1946	M. Rodrigo	S. B. Pilapitiya	3	0	Trinity (2)	Trinity	0	8	Colombo	Kandy
2	3	1947	N. W. Karunaratne	G. Sanmugam	0	0	Trinity (3)	Trinity	6	0	Kandy	Colombo
3	4	1948	A. A. Cader	A. R. A. Mohomed	6	8	Royal	Royal	3	6	Colombo	Kandy
4	5	1949	G. C. Weinman	K. Arumugam	5	8	Royal (2)	Royal	3	3	Colombo	Kandy
...
73	74	2018	S. Feroze	A. Shiek	39	27	Royal (33)	Royal	7	19	Colombo	Kandy
74	75	2019	T. Hassen	R. Bandaranayake	34	13	Royal (34)	Royal	17	24	Kandy	Colombo
75	76	2022	H. Peiris	L. Moragoda	29	21	Royal (35)	Royal	0	26	Colombo	Kandy

Next steps: [Generate code with df](#) [View recommended plots](#) [New interactive sheet](#)

```
# Get list of columns
cols = list(df.columns)

# Remove 'Cumulative' from the list
cols.remove('Winner')

# Append 'Cumulative' at the end
cols.append('Winner')

# Reorder DataFrame
df = df[cols]

# Get list of columns
cols = list(df.columns)

# Remove 'Cumulative' from the list
cols.remove('Cumulative')

# Append 'Cumulative' at the end
cols.append('Cumulative')

# Reorder DataFrame
df = df[cols]

# Get current columns
cols = list(df.columns)

# Remove the columns you want to move
cols.remove('1st leg Venue')
cols.remove('2nd leg Venue')

# Insert them at positions 5 and 6 (0-indexed, so index 4 and 5)
cols.insert(4, '1st leg Venue')
cols.insert(5, '2nd leg Venue')

# Reorder the DataFrame
df = df[cols]

# Convert leg columns to numeric and fill NaN with 0
df['Royal 1st leg'] = pd.to_numeric(df['Royal 1st leg'], errors='coerce').fillna(0).astype(int)
df['Royal 2nd leg'] = pd.to_numeric(df['Royal 2nd leg'], errors='coerce').fillna(0).astype(int)
df['Trinity 1st leg'] = pd.to_numeric(df['Trinity 1st leg'], errors='coerce').fillna(0).astype(int)
df['Trinity 2nd leg'] = pd.to_numeric(df['Trinity 2nd leg'], errors='coerce').fillna(0).astype(int)

# Compute totals as integers
df['Royal Total'] = df['Royal 1st leg'] + df['Royal 2nd leg']
df['Trinity Total'] = df['Trinity 1st leg'] + df['Trinity 2nd leg']


# Reorder columns if needed
cols = list(df.columns)
cols.remove('Royal Total')
cols.remove('Trinity Total')
```

```
cols.insert(10, 'Royal Total')
cols.insert(11, 'Trinity Total')
df = df[cols]

# Manually update row index 26
df.at[26, '1st leg Venue'] = 'Colombo'
df.at[26, '2nd leg Venue'] = 'Second leg not played'
df.at[26, 'Royal 1st leg'] = 22
df.at[26, 'Trinity 1st leg'] = 3

# Optional: update totals for this row
df.at[26, 'Royal Total'] = df.at[26, 'Royal 1st leg'] + (df.at[26, 'Royal 2nd leg'] if pd.notna(df.at[26, 'Royal 2nd leg']))
df.at[26, 'Trinity Total'] = df.at[26, 'Trinity 1st leg'] + (df.at[26, 'Trinity 2nd leg'] if pd.notna(df.at[26, 'Trinity 2nd leg']))


# Preview updated row
df.loc[26]
```



	26
Series	27
Year	1971
Royal Skipper	F. R. Perera
Trinity Skipper	Y. S. Ping
1st leg Venue	Colombo
2nd leg Venue	Second leg not played
Royal 1st leg	22
Royal 2nd leg	0
Trinity 1st leg	3
Trinity 2nd leg	0
Royal Total	22
Trinity Total	3
Winner	Royal
Cumulative	Royal (10)

dtype: object

df




	Series	Year	Royal Skipper	Trinity Skipper	1st leg Venue	2nd leg Venue	Royal 1st leg	Royal 2nd leg	Trinity 1st leg	Trinity 2nd leg	Royal Total	Trinity Total	Winner	Cumulative
0	1	1945	C. D. L. Fernando	R. G. Sourjah (1L) S. B. Pilapitiya (2L)	Colombo	Kandy	3	0	0	6	3	6	Trinity	Trinity
1	2	1946	M. Rodrigo	S. B. Pilapitiya	Colombo	Kandy	3	0	0	8	3	8	Trinity	Trinity
2	3	1947	N. W. Karunaratne	G. Sanmugam	Kandy	Colombo	0	0	6	0	0	6	Trinity	Trinity
3	4	1948	A. A. Cader	A. R. A. Mohamed	Colombo	Kandy	6	8	3	6	14	9	Royal	Royal
4	5	1949	G. C. Weinman	K. Arumugam	Colombo	Kandy	5	8	3	3	13	6	Royal	Royal
...
73	74	2018	S. Feroze	A. Shiek	Colombo	Kandy	39	27	7	19	66	26	Royal	Royal (
74	75	2019	T. Hassen	R. Bandaranayake	Kandy	Colombo	34	13	17	24	47	41	Royal	Royal (
75	76	2022	H. Peiris	L. Moragoda	Colombo	Kandy	29	21	0	26	50	26	Royal	Royal (
76	77	2023	R. Senanavake	A. Manzil	Kandy	Colombo	10	27	13	17	37	30	Royal	Royal (

Next steps: [Generate code with df](#) [View recommended plots](#) [New interactive sheet](#)

✓ Highest bradby total for Royal

```
# Get the full row with maximum Royal Total
df.loc[df['Royal Total'].idxmax()]
```




	57
Series	58
Year	2002
Royal Skipper	Z. Hamid
Trinity Skipper	N. S. A. Mendis
1st leg Venue	Colombo
2nd leg Venue	Kandy
Royal 1st leg	39
Royal 2nd leg	44
Trinity 1st leg	0
Trinity 2nd leg	0
Royal Total	83
Trinity Total	0
Winner	Royal
Cumulative	Royal (25)

dtype: object

✓ Highest Bradby Total for Trinity

```
# Get the full row with maximum Trinity Total
df.loc[df['Trinity Total'].idxmax()]
```




	67
Series	68
Year	2012
Royal Skipper	A. Jamaldeen
Trinity Skipper	K. Seneviratne
1st leg Venue	Colombo
2nd leg Venue	Kandy
Royal 1st leg	11
Royal 2nd leg	20
Trinity 1st leg	34
Trinity 2nd leg	36
Royal Total	31
Trinity Total	70
Winner	Trinity
Cumulative	Trinity (37)

dtype: object

✓ Lowest Total for Royal in Bradby

```
# Get the full row with minimum Royal Total
df.loc[df['Royal Total'].idxmin()]
```




	2
Series	3
Year	1947
Royal Skipper	N. W. Karunaratne
Trinity Skipper	G. Sanmugam
1st leg Venue	Kandy
2nd leg Venue	Colombo
Royal 1st leg	0
Royal 2nd leg	0
Trinity 1st leg	6
Trinity 2nd leg	0
Royal Total	0
Trinity Total	6
Winner	Trinity
Cumulative	Trinity (3)

dtype: object

Lowest Total for Trinity in Bradby

```
# Get the full row with minimum Royal Total
df.loc[df['Trinity Total'].idxmin()]
```




	13
Series	14
Year	1958
Royal Skipper	D. N. Fernando
Trinity Skipper	K. B. de Joedt
1st leg Venue	Colombo
2nd leg Venue	Kandy
Royal 1st leg	6
Royal 2nd leg	0
Trinity 1st leg	0
Trinity 2nd leg	0
Royal Total	6
Trinity Total	0
Winner	Royal
Cumulative	Royal (4)

dtype: object

Highest Score in a single leg for Royal


```
# Get the full row with maximum Royal 1st leg
df.loc[df['Royal 1st leg'].idxmax()]
```

	59
Series	60
Year	2004
Royal Skipper	A. L. Dissanayake
Trinity Skipper	D. Senarath
1st leg Venue	Colombo
2nd leg Venue	Kandy
Royal 1st leg	41
Royal 2nd leg	10
Trinity 1st leg	6
Trinity 2nd leg	0
Royal Total	51
Trinity Total	6
Winner	Royal
Cumulative	Royal (27)

dtype: object

```
# Get the full row with maximum Royal 2nd leg
df.loc[df['Royal 2nd leg'].idxmax()]
```



	70
Series	71
Year	2015
Royal Skipper	B. Gamage
Trinity Skipper	I. Rangala
1st leg Venue	Kandy
2nd leg Venue	Colombo
Royal 1st leg	23
Royal 2nd leg	49
Trinity 1st leg	22
Trinity 2nd leg	0
Royal Total	72
Trinity Total	22
Winner	Royal
Cumulative	Royal (31)

dtype: object

✓ Highest Score in a single leg for Trinity

```
# Get the full row with maximum Trinity 1st leg
df.loc[df['Trinity 1st leg'].idxmax()]
```



	65
Series	66
Year	2010
Royal Skipper	D. Attygalle
Trinity Skipper	D. Dissanayake
1st leg Venue	Colombo
2nd leg Venue	Kandy
Royal 1st leg	38
Royal 2nd leg	34
Trinity 1st leg	37
Trinity 2nd leg	17
Royal Total	72
Trinity Total	54
Winner	Royal
Cumulative	Royal (30)

dtype: object

```
# Get the full row with maximum Trinity 2nd leg
df.loc[df['Trinity 2nd leg'].idxmax()]
```




	66
Series	67
Year	2011
Royal Skipper	S. Pathirana
Trinity Skipper	M. Ramzeen
1st leg Venue	Kandy
2nd leg Venue	Colombo
Royal 1st leg	33
Royal 2nd leg	5
Trinity 1st leg	25
Trinity 2nd leg	40
Royal Total	38
Trinity Total	65
Winner	Trinity
Cumulative	Trinity (36)

dtype: object

Summary

```
df[['Royal Total','Trinity Total', 'Royal 1st leg','Trinity 1st leg', 'Royal 2nd leg','Trinity 2nd leg']].describe()
```



	Royal Total	Trinity Total	Royal 1st leg	Trinity 1st leg	Royal 2nd leg	Trinity 2nd leg
count	78.000000	78.000000	78.000000	78.000000	78.000000	78.000000
mean	21.935897	20.205128	11.448718	10.589744	10.487179	9.615385
std	19.368162	16.586929	11.122281	9.867217	10.629222	8.964911
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	6.500000	6.000000	3.000000	3.000000	3.000000	3.000000
50%	15.000000	16.500000	6.000000	7.000000	8.000000	8.000000
75%	34.000000	27.750000	18.750000	17.000000	13.750000	12.750000
max	83.000000	70.000000	41.000000	37.000000	49.000000	40.000000



```

import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Prepare data in long format for each matchup
df_total = df[['Royal Total', 'Trinity Total']].melt(var_name='Team', value_name='Score')
df_first = df[['Royal 1st leg', 'Trinity 1st leg']].melt(var_name='Team', value_name='Score')
df_second = df[['Royal 2nd leg', 'Trinity 2nd leg']].melt(var_name='Team', value_name='Score')

# Plot 3 subplots
fig, axes = plt.subplots(1, 3, figsize=(15,6), sharey=True)

sns.boxplot(data=df_total, x='Team', y='Score', palette=['blue','red'], ax=axes[0])
axes[0].set_title("Total Scores")

sns.boxplot(data=df_first, x='Team', y='Score', palette=['blue','red'], ax=axes[1])
axes[1].set_title("1st Leg Scores")

sns.boxplot(data=df_second, x='Team', y='Score', palette=['blue','red'], ax=axes[2])
axes[2].set_title("2nd Leg Scores")

# General formatting
for ax in axes:
    ax.set_xlabel("")
    ax.set_ylabel("Score")
    ax.grid(axis='y', linestyle='--', alpha=0.3)

plt.suptitle("Royal vs Trinity Boxplots", fontsize=16)
plt.tight_layout()
plt.show()

```

↗ /tmp/ipython-input-1440637193.py:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue`

```

sns.boxplot(data=df_total, x='Team', y='Score', palette=['blue','red'], ax=axes[0])
/tmp/ipython-input-1440637193.py:16: FutureWarning:

```

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue`

```

sns.boxplot(data=df_first, x='Team', y='Score', palette=['blue','red'], ax=axes[1])
/tmp/ipython-input-1440637193.py:19: FutureWarning:

```

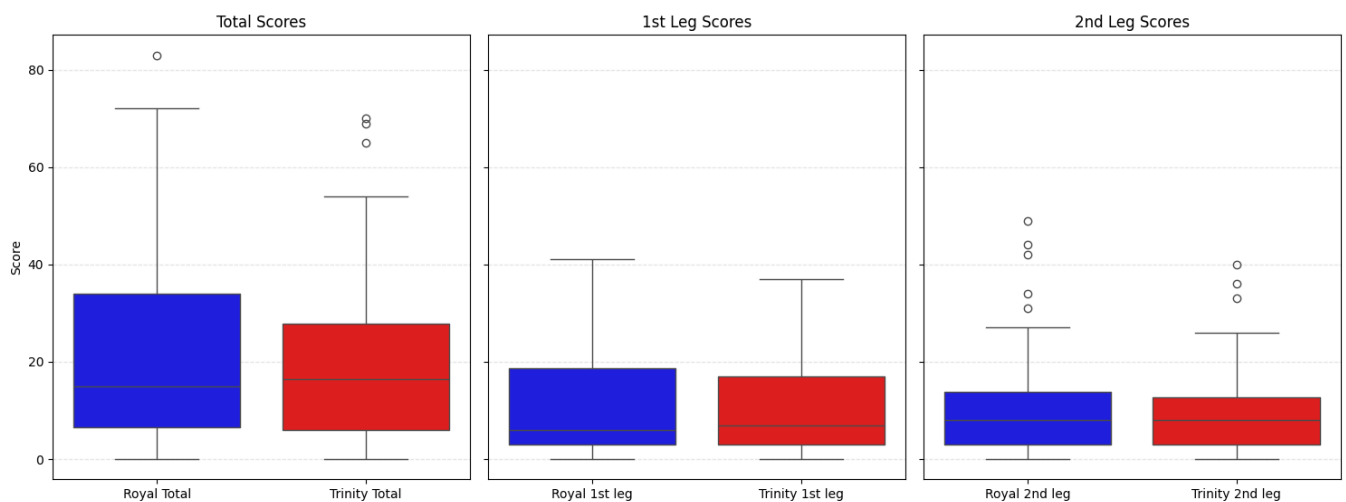
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue`

```

sns.boxplot(data=df_second, x='Team', y='Score', palette=['blue','red'], ax=axes[2])

```

Royal vs Trinity Boxplots



Example: your existing DataFrame is df

```

# 1. Show unique values before standardization
print("Unique values in 'Winner' before standardization:")
print(df['Winner'].unique())

```

```

# 2. Standardize all variations of draw (anything containing 'draw', case-insensitive)

```

```
df['Winner'] = df['Winner'].apply(lambda x: 'Draw' if 'draw' in str(x).lower() else x)
```

```
# 3. Show unique values after standardization
print("\nUnique values in 'Winner' after standardization:")
print(df['Winner'].unique())
```

```
Unique values in 'Winner' before standardization:
['Trinity' 'Royal' 'Draw' 'Draw (2)']
```

```
Unique values in 'Winner' after standardization:
['Trinity' 'Royal' 'Draw']
```

```
print(df['Winner'].unique())
```

```
['Trinity' 'Royal' 'Draw']
```

```
import matplotlib.pyplot as plt
```

```
# Count number of wins per team
win_counts = df['Winner'].value_counts()
```

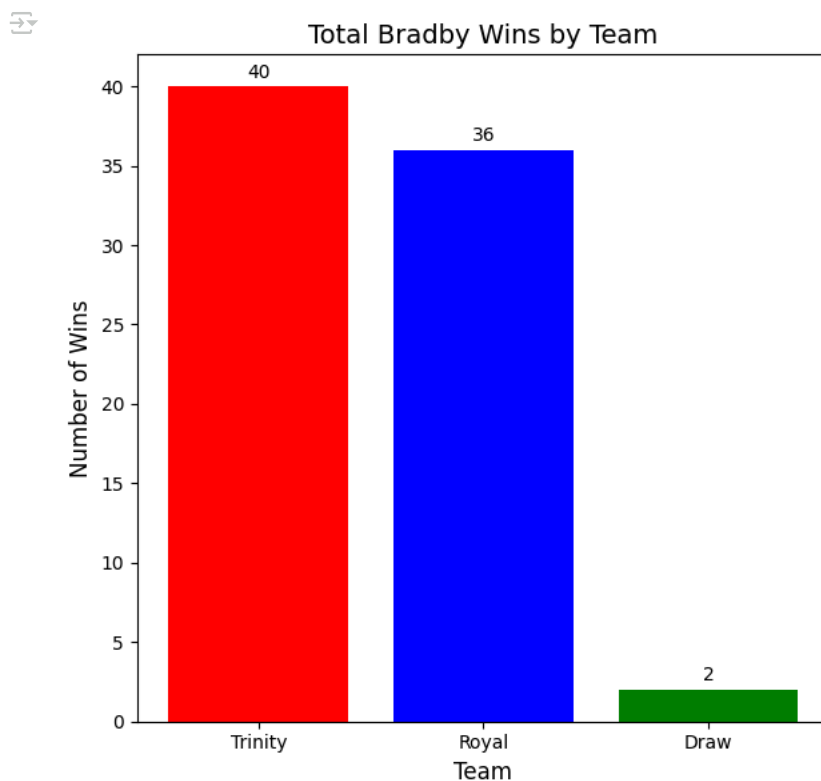
```
# Set colors: Draw = green, Royal = blue, Trinity = red
colors = ['green' if team.lower() == 'draw' else ('blue' if team.lower() == 'royal' else 'red')
          for team in win_counts.index]
```

```
# Taller figure
plt.figure(figsize=(6,6)) # width=6, height=6
```

```
bars = plt.bar(win_counts.index, win_counts.values, color=colors)
```

```
# Add labels on top of bars
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, height + 0.3, int(height), ha='center', va='bottom', fontsize=10)
```

```
plt.title('Total Bradby Wins by Team', fontsize=14)
plt.xlabel('Team', fontsize=12)
plt.ylabel('Number of Wins', fontsize=12)
plt.xticks(rotation=0)
plt.ylim(0, win_counts.max() + 2) # ensure labels fit above bars
plt.tight_layout()
plt.show()
```




```
# Ensure numeric columns
numeric_cols = ['Royal 1st leg', 'Royal 2nd leg', 'Trinity 1st leg', 'Trinity 2nd leg']
for col in numeric_cols:
    df[col] = pd.to_numeric(df[col], errors='coerce').fillna(0)

# Royal won both legs and Bradby
royal_double_win = df[
    (df['Royal 1st leg'] > df['Trinity 1st leg']) &
    (df['Royal 2nd leg'] > df['Trinity 2nd leg']) &
    (df['Winner'].str.lower() == 'royal')
]

# Trinity won both legs and Bradby
trinity_double_win = df[
    (df['Trinity 1st leg'] > df['Royal 1st leg']) &
    (df['Trinity 2nd leg'] > df['Royal 2nd leg']) &
    (df['Winner'].str.lower() == 'trinity')
]

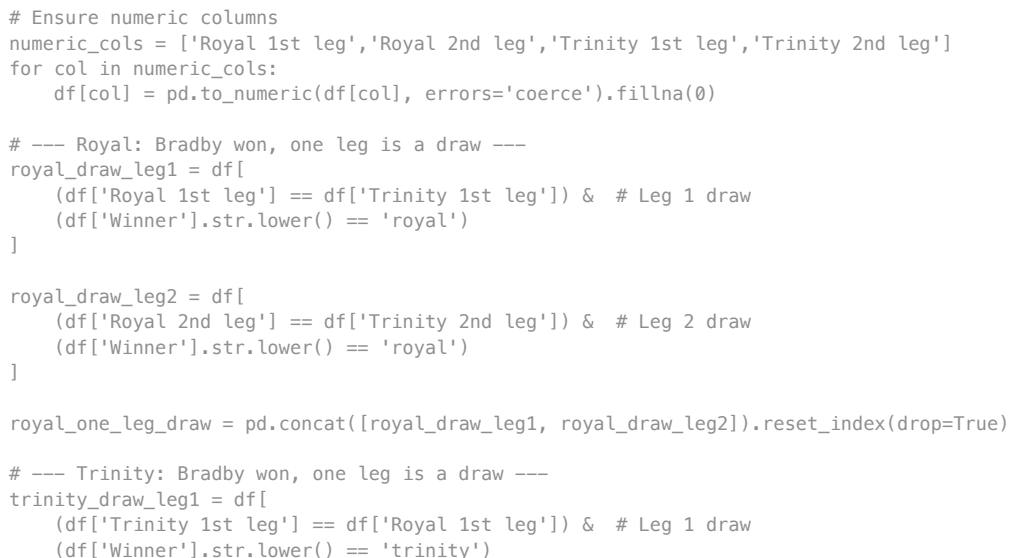
# Combine if needed
double_leg_wins = pd.concat([royal_double_win, trinity_double_win])

double_leg_wins
```



	Series	Year	Royal Skipper	Trinity Skipper	1st leg Venue	2nd leg Venue	Royal 1st leg	Royal 2nd leg	Trinity 1st leg	Trinity 2nd leg	Royal Total	Trinity Total	Winner	Ci
3	4	1948	A. A. Cader	A. R. A. Mohomed	Colombo	Kandy	6	8	3	6	14	9	Royal	
4	5	1949	G. C. Weinman	K. Arumugam	Colombo	Kandy	5	8	3	3	13	6	Royal	
6	7	1951	H. E. Wijesinghe	M. S. Panditharatne	Colombo	Kandy	19	13	3	5	32	8	Royal	
14	15	1959	M. L. Anghie	D. L. Kobbekaduwa	Kandy	Colombo	5	6	0	0	11	0	Royal	
16	17	1961	J. V. P. Samarasekara	J. W. Jayawardena	Kandy	Colombo	6	8	3	0	14	3	Royal	
19	20	1964	K. D. T. Paul	M. T. Sahayam	Colombo	Kandy	3	14	0	6	17	6	Royal	
23	24	1968	C. R. de Silva	A. L. Abeyratne	Colombo	Kandy	19	5	0	3	24	3	Royal	
30	31	1975	H. A. I. Hassen	M. R. Sourjah	Kandy	Colombo	13	21	11	3	34	14	Royal	
31	32	1976	M. Weerakumar	S. V. Ranasinghe	Colombo	Kandy	36	25	0	6	61	6	Royal	
35	36	1980	S. Sujanthakumar	T. B. Ellepola	Colombo	Kandy	7	16	3	3	23	6	Royal	
43	44	1988	L. W. de Z. Gunarathne	B. Dandeniya	Colombo	Kandy	24	13	0	0	37	0	Royal	
44	45	1989	J. H. A. Dhammika	R. Ranaraja	Kandy	Colombo	9	9	6	8	18	14	Royal	
48	49	1993	C. R. A. Abeysuriya	N. Muhandiramge	Kandy	Colombo	3	10	0	3	13	3	Royal	
56	57	2001	H. Kaluaratchi	T. Jayawardena	Kandy	Colombo	12	16	6	0	28	6	Royal	
57	58	2002	Z. Hamid	N. S. A. Mendis	Colombo	Kandy	39	44	0	0	83	0	Royal	
58	59	2003	R. Jayasundara	M. Maddumapatabendi	Kandy	Colombo	13	13	11	8	26	19	Royal	
59	60	2004	A. L. Dissanayake	D. Senarath	Colombo	Kandy	41	10	6	0	51	6	Royal	
62	63	2007	M. Abeyratne	M. Wickramasiri (1L) S. Wanigasekara (2L)	Kandy	Colombo	20	15	12	11	35	23	Royal	
64	65	2009	N. Dhason	V. Wijesinghe	Kandy	Colombo	22	31	12	15	53	27	Royal	
65	66	2010	D. Attygalle	D. Dissanayake	Colombo	Kandy	38	34	37	17	72	54	Royal	
70	71	2015	B. Gamage	I. Rangala	Kandy	Colombo	23	49	22	0	72	22	Royal	
72	73	2017	O. Askey	N. Yee	Kandy	Colombo	22	13	17	8	35	25	Royal	
73	74	2018	S. Feroze	A. Shiek	Colombo	Kandy	39	27	7	19	66	26	Royal	
5	6	1950	T. St. C. Anghie	S. S. Bambaradeniya	Kandy	Colombo	0	3	6	5	3	11	Trinity	
7	8	1952	A. B. Van Twest	M. S. Panditharatne	Colombo	Kandy	0	0	6	12	0	18	Trinity	
8	9	1953	T. S. Almeida (1L) T. L. K. Mendis (2L)	D. Madugalle	Kandy	Colombo	0	0	13	3	0	16	Trinity	
9	10	1954	K. W. W. Edwards	L. L. Vitharana	Colombo	Kandy	0	5	5	6	5	11	Trinity	
11	12	1956	T. L. Almeida	D. N. Frank	Colombo	Kandy	0	0	15	11	0	26	Trinity	
12	13	1957	R. Sivaratnam	K. B. de Joedt	Kandy	Colombo	0	8	8	9	8	17	Trinity	
15	16	1960	H. S. de Silva	E. D. K. Roles	Colombo	Kandy	3	0	5	8	3	13	Trinity	
22	23	1967	D. J. Perera (1L) N. B. L. Lieversz (2L)	A. L. Abeyratne	Kandy	Colombo	3	3	17	16	6	33	Trinity	
24	25	1969	M. H. C. Malwatte	M. S. Jainudeen	Kandy	Colombo	0	3	8	9	3	17	Trinity	
25	26	1970	C. J. Fernando	S. P. Samarasekara	Colombo	Kandy	3	12	19	16	15	35	Trinity	
29	30	1974	M. B. Akbar	C. Y. Ching	Colombo	Kandy	3	9	18	12	12	30	Trinity	
32	33	1977	S. S. Hashim	R. N. Balasuriya	Kandy	Colombo	6	4	12	10	10	22	Trinity	
37	38	1982	H. Muttiah	B. S. J. Fernando	Colombo	Kandy	6	4	9	6	10	15	Trinity	
41	42	1986	M. J. Peiris	S. N. Gunaratne	Colombo	Kandy	3	3	6	9	6	15	Trinity	

Next steps: [Generate code with double_leg_wins](#) [View recommended plots](#) [New interactive sheet](#)




```
]

trinity_draw_leg2 = df[
    (df['Trinity 2nd leg'] == df['Royal 2nd leg']) & # Leg 2 draw
    (df['Winner'].str.lower() == 'trinity')
]

trinity_one_leg_draw = pd.concat([trinity_draw_leg1, trinity_draw_leg2]).reset_index(drop=True)

# Combine both teams
one_leg_draw_bradby = pd.concat([royal_one_leg_draw, trinity_one_leg_draw]).reset_index(drop=True)

one_leg_draw_bradby
```



	Series	Year	Royal Skipper	Trinity Skipper	1st leg Venue	2nd leg Venue	Royal 1st leg	Royal 2nd leg	Trinity 1st leg	Trinity 2nd leg	Royal Total	Trinity Total	Winner	Cumulativ
0	34	1978	K. R. T. Peiris	J. R. Kiridena	Colombo	Kandy	0	8	0	4	8	4	Royal	Royal (1
1	14	1958	D. N. Fernando	K. B. de Joedt	Colombo	Kandy	6	0	0	0	6	0	Royal	Royal (
2	21	1965	U. P. Wickramasinghe	M. T. M. Zarook	Kandy	Colombo	8	3	3	3	11	6	Royal	Royal (
3	27	1971	F. R. Perera	Y. S. Ping	Colombo	Second leg not played	22	0	3	0	22	3	Royal	Royal (1
4	40	1984	S. Agalawatte	D. M. Wijesinghe	Colombo	Kandy	6	0	3	0	6	3	Royal	Royal (1
5	50	1994	R. Malalasekara	A. M. Rally	Colombo	Kandy	10	8	3	8	18	11	Royal	Royal (2
6	19	1963	M. Jayakumar	B. D. A. Piyasena	Kandy	Colombo	6	3	6	5	9	11	Trinity	Trinity (1
7	22	1966	I. R. Thurairatnam	E. G. Van Langenberg	Colombo	Kandy	3	3	3	8	6	11	Trinity	Trinity (1

Next steps: [Generate code with one_leg_draw_bradby](#) [View recommended plots](#) [New interactive sheet](#)

```
import matplotlib.pyplot as plt

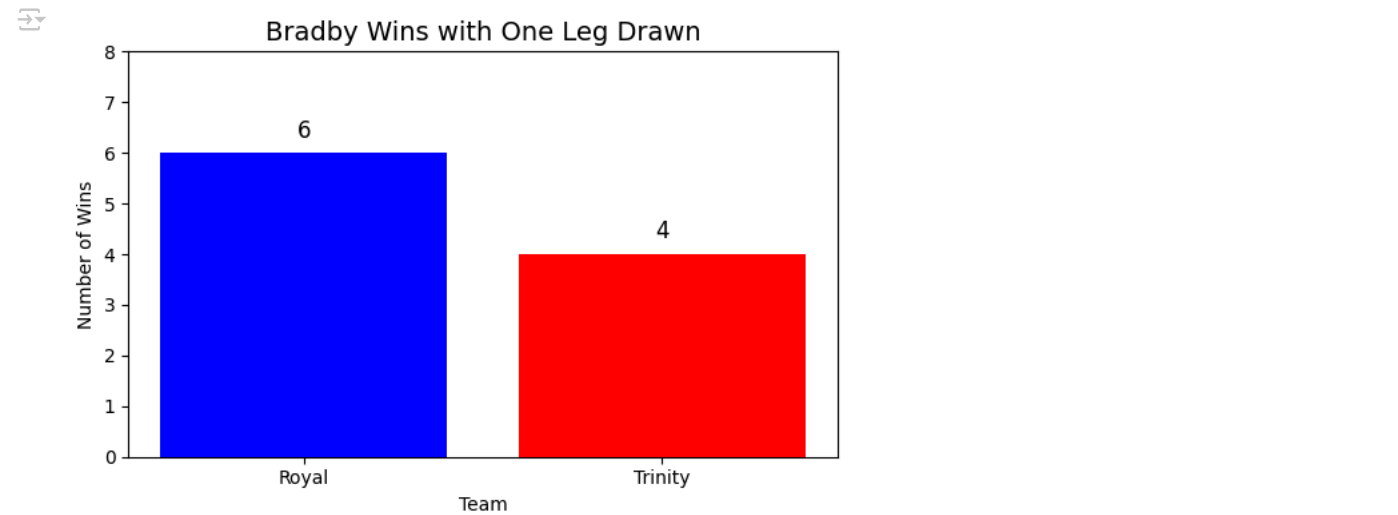
# Count the number of occurrences per team
counts = one_leg_draw_bradby['Winner'].value_counts()

# Set colors
colors = ['blue' if team.lower()=='royal' else 'red' for team in counts.index]

# Plot
plt.figure(figsize=(6,4))
bars = plt.bar(counts.index, counts.values, color=colors)

# Add value labels on top of bars
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, height + 0.2, int(height),
             ha='center', va='bottom', fontsize=12)

plt.title('Bradby Wins with One Leg Drawn', fontsize=14)
plt.ylabel('Number of Wins')
plt.xlabel('Team')
plt.ylim(0, counts.max() + 2)
plt.tight_layout()
plt.show()
```

```
import pandas as pd

# Example: df is your DataFrame

# --- Royal won 1st leg, lost 2nd leg, but won overall ---
royal_first_leg_win = df[(df['Royal 1st leg'] > df['Trinity 1st leg']) &
                          (df['Royal 2nd leg'] < df['Trinity 2nd leg']) &
                          (df['Winner'].str.lower() == 'royal')]

# --- Trinity won 1st leg, lost 2nd leg, but won overall ---
trinity_first_leg_win = df[(df['Trinity 1st leg'] > df['Royal 1st leg']) &
                            (df['Trinity 2nd leg'] < df['Royal 2nd leg']) &
                            (df['Winner'].str.lower() == 'trinity')]

# --- Combine both ---
first_leg_win_comebacks = pd.concat([royal_first_leg_win, trinity_first_leg_win]).reset_index(drop=True)
```

first_leg_win_comebacks

	Series	Year	Royal Skipper	Trinity Skipper	1st leg Venue	2nd leg Venue	Royal 1st leg	Royal 2nd leg	Trinity 1st leg	Trinity 2nd leg	Royal Total	Trinity Total	Winner	Cumulat:
0	29	1973	M. G. Muller	J. K. Yu	Kandy	Colombo	18	4	4	12	22	16	Royal	Royal
1	46	1990	R. Jayasuriya	W. D. S. Premaratne (1L) R. Kalpage (2L)	Colombo	Kandy	23	0	8	6	23	14	Royal	Royal
2	75	2019	T. Hassen	R. Bandaranayake	Kandy	Colombo	34	13	17	24	47	41	Royal	Royal
3	76	2022	H. Peiris	L. Moragoda	Colombo	Kandy	29	21	0	26	50	26	Royal	Royal
4	37	1981	N. S. Cooray	R. Bandaranayake	Kandy	Colombo	3	10	16	3	13	19	Trinity	Trinity
5	39	1983	S. Cooray	A. D. Ratwatte	Kandy	Colombo	6	10	14	6	16	20	Trinity	Trinity
6	41	1985	C. Nanayakkara	M. Jayatissa	Kandy	Colombo	9	4	18	3	13	21	Trinity	Trinity
7	53	1997	A. Yusuf	N. S. Fernando	Kandy	Colombo	6	18	14	11	24	25	Trinity	Trinity
8	56	2000	S. Perera	H. A. Luchow	Colombo	Kandy	25	20	32	17	45	49	Trinity	Trinity

Next steps: [Generate code with first_leg_win_comebacks](#) [View recommended plots](#) [New interactive sheet](#)


```
# Royal won 1st leg but lost 2nd leg, still won Bradby
royal_first_leg_comeback = df[
    (df['Royal 1st leg'] > df['Trinity 1st leg']) &
    (df['Royal 2nd leg'] < df['Trinity 2nd leg']) &
    (df['Winner'].str.lower() == 'royal')
]

# Trinity won 1st leg but lost 2nd leg, still won Bradby
trinity_first_leg_comeback = df[
```

```
(df['Trinity 1st leg'] > df['Royal 1st leg']) &
(df['Trinity 2nd leg'] < df['Royal 2nd leg']) &
(df['Winner'].str.lower() == 'trinity')
]

first_leg_comebacks = pd.concat([royal_first_leg_comeback, trinity_first_leg_comeback])
```

first_leg_comebacks



	Series	Year	Royal Skipper	Trinity Skipper	1st leg Venue	2nd leg Venue	Royal 1st leg	Royal 2nd leg	Trinity 1st leg	Trinity 2nd leg	Royal Total	Trinity Total	Winner	Cumulative
28	29	1973	M. G. Muller	J. K. Yu	Kandy	Colombo	18	4	4	12	22	16	Royal	Royal
45	46	1990	R. Jayasuriya	W. D. S. Premaratne (1L) R. Kalpage (2L)	Colombo	Kandy	23	0	8	6	23	14	Royal	Royal
74	75	2019	T. Hassen	R. Bandaranayake	Kandy	Colombo	34	13	17	24	47	41	Royal	Royal
75	76	2022	H. Peiris	L. Moragoda	Colombo	Kandy	29	21	0	26	50	26	Royal	Royal
36	37	1981	N. S. Cooray	R. Bandaranayake	Kandy	Colombo	3	10	16	3	13	19	Trinity	Trinity
38	39	1983	S. Cooray	A. D. Ratwatte	Kandy	Colombo	6	10	14	6	16	20	Trinity	Trinity
40	41	1985	C. Nanayakkara	M. Jayatissa	Kandy	Colombo	9	4	18	3	13	21	Trinity	Trinity
52	53	1997	A. Yusuf	N. S. Fernando	Kandy	Colombo	6	18	14	11	24	25	Trinity	Trinity
55	56	2000	S. Perera	H. A. Luchow	Colombo	Kandy	25	20	32	17	45	49	Trinity	Trinity

Next steps: [Generate code with first_leg_comebacks](#) [View recommended plots](#) [New interactive sheet](#)

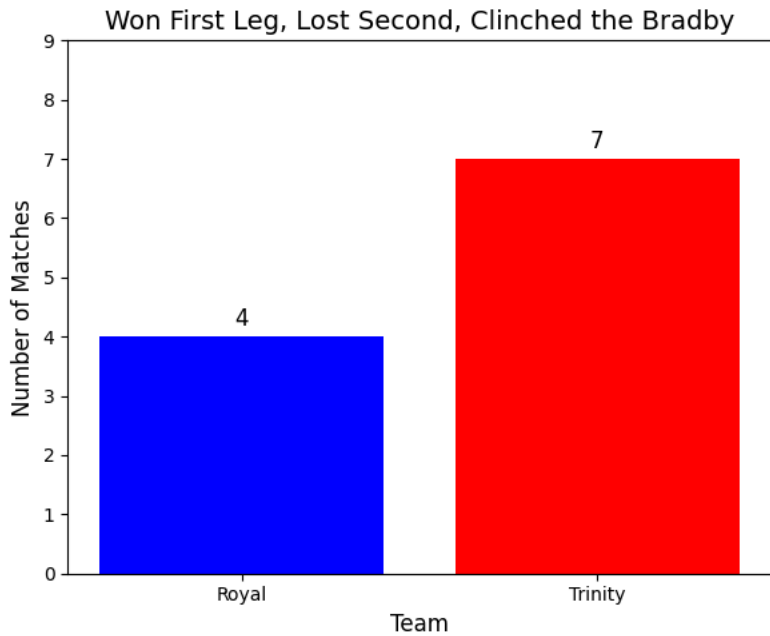
```
# Count of matches for each team
royal_count = len(royal_first_leg_win)
trinity_count = len(trinity_first_leg_win)

teams = ['Royal', 'Trinity']
counts = [royal_count, trinity_count]

# Plot bar chart
plt.figure(figsize=(6,5))
bars = plt.bar(teams, counts, color=['blue', 'red'])

# Add labels on top of bars
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, height + 0.1, str(height),
             ha='center', va='bottom', fontsize=12)

plt.title('Won First Leg, Lost Second, Clinched the Bradby', fontsize=14)
plt.ylabel('Number of Matches', fontsize=12)
plt.xlabel('Team', fontsize=12)
plt.ylim(0, max(counts) + 2)
plt.tight_layout()
plt.show()
```



```
# Royal lost 1st leg but won the overall match
royal_comeback = df[(df['Royal 1st leg'] < df['Trinity 1st leg']) & (df['Winner'].str.lower() == 'royal')]

# Trinity lost 1st leg but won the overall match
trinity_comeback = df[(df['Trinity 1st leg'] < df['Royal 1st leg']) & (df['Winner'].str.lower() == 'trinity')]

# Combine both if you want a single DataFrame
comeback_matches = pd.concat([royal_comeback, trinity_comeback])

# Preview
comeback_matches
```



	Series	Year	Royal Skipper	Trinity Skipper	1st leg Venue	2nd leg Venue	Royal 1st leg	Royal 2nd leg	Trinity 1st leg	Trinity 2nd leg	Royal Total	Trinity Total	Winner	Cumulative
34	35	1979	R. P. Gunasekara	J. V. Tissera	Kandy	Colombo	3	18	4	0	21	4	Royal	Royal (15)
53	54	1998	W. S. S. R. Perera	P. Jayawardena	Colombo	Kandy	18	42	20	11	60	31	Royal	Royal (23)
76	77	2023	R. Senanayake	A. Manzil	Kandy	Colombo	10	27	13	17	37	30	Royal	Royal (36)
0	1	1945	C. D. L. Fernando	R. G. Sourjah (1L) S. B. Pilapitiya (2L)	Colombo	Kandy	3	0	0	6	3	6	Trinity	Trinity
1	2	1946	M. Rodrigo	S. B. Pilapitiya	Colombo	Kandy	3	0	0	8	3	8	Trinity	Trinity (2)
17	18	1962	U. L. Kaluaratchi	N. T. E. Brohier	Colombo	Kandy	5	0	0	9	5	9	Trinity	Trinity (12)

Next steps:

[Generate code with comeback_matches](#)
[View recommended plots](#)
[New interactive sheet](#)

```
import matplotlib.pyplot as plt

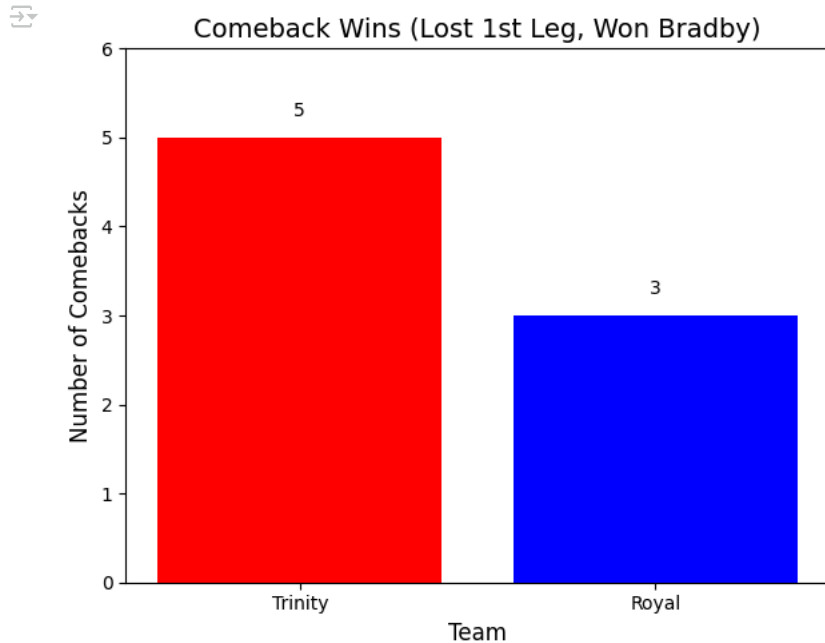
# Count comebacks per team
comeback_counts = comeback_matches['Winner'].value_counts()

# Plot bar chart
plt.figure(figsize=(6,5)) # slightly taller figure
bars = plt.bar(comeback_counts.index, comeback_counts.values, color=['red', 'blue'])

# Add labels on top of bars
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, height + 0.2, int(height), ha='center', va='bottom', fontsize=10)

# Increase y-axis limit slightly above max value
plt.ylim(0, comeback_counts.max() + 1)
```

```
plt.title('Comeback Wins (Lost 1st Leg, Won Bradby)', fontsize=14)
plt.xlabel('Team', fontsize=12)
plt.ylabel('Number of Comebacks', fontsize=12)
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```



```
# Make a copy to avoid SettingWithCopyWarning
comeback_matches = comeback_matches.copy()


# First Leg Gap
comeback_matches['First Leg Gap'] = comeback_matches.apply(
    lambda row: row['Trinity 1st leg'] - row['Royal 1st leg']
    if row['Winner'].lower() == 'royal'
    else row['Royal 1st leg'] - row['Trinity 1st leg'],
    axis=1
)

# Second Leg Gap
comeback_matches['Second Leg Gap'] = comeback_matches.apply(
    lambda row: row['Royal 2nd leg'] - row['Trinity 2nd leg']
    if row['Winner'].lower() == 'royal'
    else row['Trinity 2nd leg'] - row['Royal 2nd leg'],
    axis=1
)



# Total Bradby Gap (sum of both legs)
comeback_matches['Bradby Gap'] = comeback_matches.apply(
    lambda row: row['Royal Total'] - row['Trinity Total'] if row['Winner'].lower() == 'royal'
    else row['Trinity Total'] - row['Royal Total'],
    axis=1
)

# Sort by first leg gap if you like
comeback_matches_sorted = comeback_matches.sort_values(by='First Leg Gap', ascending=False)

# Preview relevant columns
comeback_matches_sorted[['Year', 'Winner', 'Royal 1st leg', 'Trinity 1st leg',
                        'First Leg Gap', 'Royal 2nd leg', 'Trinity 2nd leg',
                        'Second Leg Gap', 'Bradby Gap']]
```



	Year	Winner	Royal 1st leg	Trinity 1st leg	First Leg Gap	Royal 2nd leg	Trinity 2nd leg	Second Leg Gap	Bradby Gap
66	2011	Trinity	33	25	8	5	40	35	27
17	1962	Trinity	5	0	5	0	9	9	4
0	1945	Trinity	3	0	3	0	6	6	3
76	2023	Royal	10	13	3	27	17	10	7
1	1946	Trinity	3	0	3	0	8	8	5
53	1998	Royal	18	20	2	42	11	31	29
34	1979	Royal	3	4	1	18	0	18	17
27	1978	Trinity	0	0	0	0	10	7	0



```
# Make a copy to avoid SettingWithCopyWarning
comeback_matches = comeback_matches.copy()


# First Leg Gap
comeback_matches['First Leg Gap'] = comeback_matches.apply(
    lambda row: row['Trinity 1st leg'] - row['Royal 1st leg']
    if row['Winner'].lower() == 'royal'
    else row['Royal 1st leg'] - row['Trinity 1st leg'],
    axis=1
)

# Second Leg Gap
comeback_matches['Second Leg Gap'] = comeback_matches.apply(
    lambda row: row['Royal 2nd leg'] - row['Trinity 2nd leg']
    if row['Winner'].lower() == 'royal'
    else row['Trinity 2nd leg'] - row['Royal 2nd leg'],
    axis=1
)



# Total Bradby Gap (sum of both legs)
comeback_matches['Bradby Gap'] = comeback_matches.apply(
    lambda row: row['Royal Total'] - row['Trinity Total'] if row['Winner'].lower() == 'royal'
    else row['Trinity Total'] - row['Royal Total'],
    axis=1
)

# Sort by first leg gap if you like
comeback_matches_sorted = comeback_matches.sort_values(by='First Leg Gap', ascending=False)

# Preview relevant columns including only 1st leg venue
comeback_matches_sorted[['Year','Winner','1st leg Venue',
    'Royal 1st leg','Trinity 1st leg','First Leg Gap',
    'Royal 2nd leg','Trinity 2nd leg','Second Leg Gap','Bradby Gap']]
```



	Year	Winner	1st leg Venue	Royal 1st leg	Trinity 1st leg	First Leg Gap	Royal 2nd leg	Trinity 2nd leg	Second Leg Gap	Bradby Gap
66	2011	Trinity	Kandy	33	25	8	5	40	35	27
17	1962	Trinity	Colombo	5	0	5	0	9	9	4
0	1945	Trinity	Colombo	3	0	3	0	6	6	3
76	2023	Royal	Kandy	10	13	3	27	17	10	7
1	1946	Trinity	Colombo	3	0	3	0	8	8	5
53	1998	Royal	Colombo	18	20	2	42	11	31	29
34	1979	Royal	Kandy	3	4	1	18	0	18	17
27	1978	Trinity	Colombo	0	0	0	0	10	7	0



```
import matplotlib.pyplot as plt

# Sort comebacks by First Leg Gap descending
comeback_sorted = comeback_matches.sort_values(by='First Leg Gap', ascending=False)

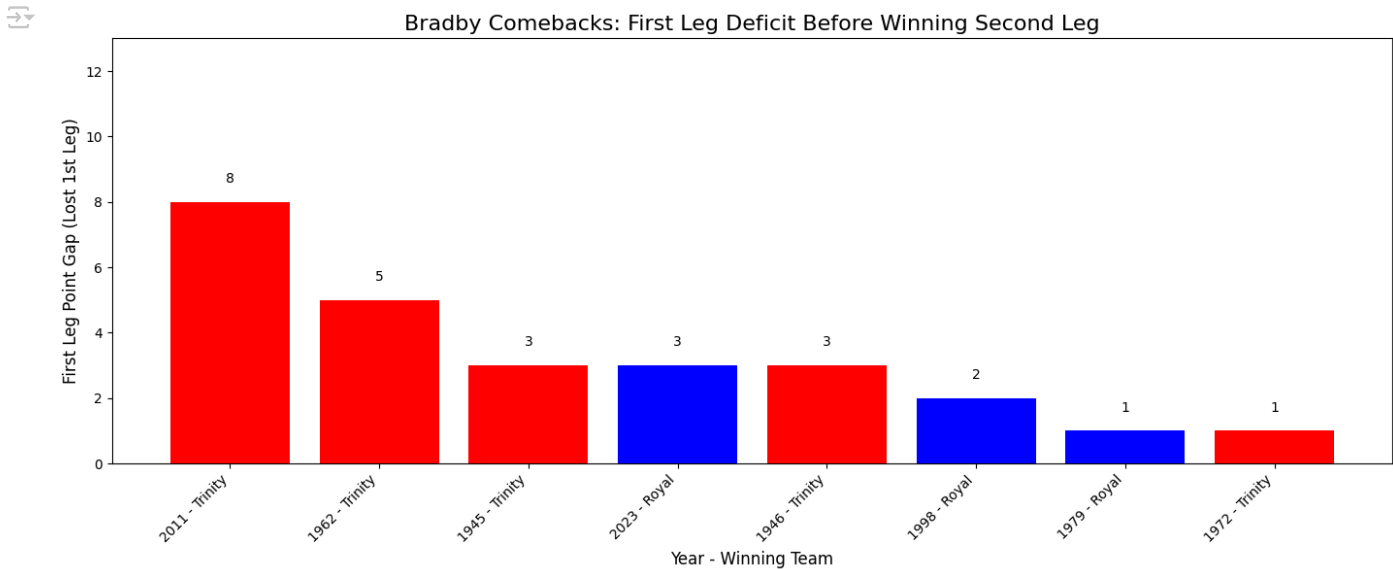
# Slightly shorter figure
plt.figure(figsize=(14,6)) # width=14, height=6

bars = plt.bar(
    comeback_sorted['Year'].astype(str) + " - " + comeback_sorted['Winner'],
    comeback_sorted['First Leg Gap'],
    color=['red' if winner.lower() == 'trinity' else 'blue' for winner in comeback_sorted['Winner']]
)
```

```
# Add labels on top of bars
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, height + 0.5, int(height), ha='center', va='bottom', fontsize=10)

# Increase y-axis limit slightly above max
plt.ylim(0, comeback_sorted['First Leg Gap'].max() + 5)

plt.title('Bradby Comebacks: First Leg Deficit Before Winning Second Leg', fontsize=16)
plt.xlabel('Year - Winning Team', fontsize=12)
plt.ylabel('First Leg Point Gap (Lost 1st Leg)', fontsize=12)
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.tight_layout()
plt.show()
```



```
# Top 10 highest Royal 1st leg scores with Year, Trinity 1st leg, and Bradby Winner
top_5_royal_1st_leg = df[['Year', 'Royal 1st leg', 'Trinity 1st leg', 'Winner']].sort_values(
    by='Royal 1st leg', ascending=False
).head(10)

top_5_royal_1st_leg
```

	Year	Royal 1st leg	Trinity 1st leg	Winner
59	2004	41	6	Royal
57	2002	39	0	Royal
73	2018	39	7	Royal
65	2010	38	37	Royal
31	1976	36	0	Royal
74	2019	34	17	Royal
66	2011	33	25	Trinity
75	2022	29	0	Royal
68	2013	25	37	Trinity
55	2000	25	32	Trinity

Next steps: [Generate code with top_5_royal_1st_leg](#) [View recommended plots](#) [New interactive sheet](#)

```
# Top 10 highest Royal 2nd leg scores with Year, Trinity 2nd leg, and Bradby Winner
top_10_royal_2nd_leg = df[['Year', 'Royal 2nd leg', 'Trinity 2nd leg', 'Winner']].sort_values(
    by='Royal 2nd leg', ascending=False
).head(10)
```

top_10_royal_2nd_leg

	Year	Royal 2nd leg	Trinity 2nd leg	Winner
70	2015	49	0	Royal
57	2002	44	0	Royal
53	1998	42	11	Royal
65	2010	34	17	Royal
64	2009	31	15	Royal
73	2018	27	19	Royal
76	2023	27	17	Royal
31	1976	25	6	Royal
77	2024	23	25	Trinity
30	1975	21	3	Royal

Next steps: [Generate code with top_10_royal_2nd_leg](#) [View recommended plots](#) [New interactive sheet](#)

```
# Top 10 highest Trinity 1st leg scores with Year, Royal 1st leg, and Bradby Winner
top_5_trinity_1st_leg = df[['Year', 'Trinity 1st leg', 'Royal 1st leg', 'Winner']].sort_values(
    by='Trinity 1st leg', ascending=False
).head(10)
```

top_5_trinity_1st_leg

	Year	Trinity 1st leg	Royal 1st leg	Winner
68	2013	37	25	Trinity
65	2010	37	38	Royal
60	2005	36	6	Trinity
67	2012	34	11	Trinity
55	2000	32	25	Trinity
51	1996	30	3	Trinity
69	2014	28	16	Trinity
66	2011	25	33	Trinity
63	2008	24	12	Trinity
70	2015	22	23	Royal

Next steps: [Generate code with top_5_trinity_1st_leg](#) [View recommended plots](#) [New interactive sheet](#)

```
# Top 10 highest Trinity 2nd leg scores with Year, Royal 2nd leg, and Bradby Winner
top_10_trinity_2nd_leg = df[['Year', 'Trinity 2nd leg', 'Royal 2nd leg', 'Winner']].sort_values(
    by='Trinity 2nd leg', ascending=False
).head(10)
```

top_10_trinity_2nd_leg

	Year	Trinity 2nd leg	Royal 2nd leg	Winner
66	2011	40	5	Trinity
67	2012	36	20	Trinity
60	2005	33	7	Trinity
61	2006	26	5	Trinity
75	2022	26	21	Royal
77	2024	25	23	Trinity
46	1991	25	10	Trinity
74	2019	24	13	Royal
50	1995	22	6	Trinity
42	1987	19	3	Trinity

Next steps: [Generate code with top_10_trinity_2nd_leg](#) [View recommended plots](#) [New interactive sheet](#)

Location

```
# Filter rows where first leg venue is Kandy and winner is Trinity
trinity_kandy_first_leg = df[(df['1st leg Venue'].str.lower() == 'kandy') &
                             (df['Winner'].str.lower() == 'trinity')]
```

```
# Preview
trinity_kandy_first_leg
```

↻

	Series	Year	Royal Skipper	Trinity Skipper	1st leg Venue	2nd leg Venue	Royal 1st leg	Royal 2nd leg	Trinity 1st leg	Trinity 2nd leg	Royal Total	Trinity Total	Winner	Cumulative
2	3	1947	N. W. Karunaratne	G. Sanmugam	Kandy	Colombo	0	0	6	0	0	6	Trinity	Trinity (6)
5	6	1950	T. St. C. Anghie	S. S. Bambaradeniya	Kandy	Colombo	0	3	6	5	3	11	Trinity	Trinity (11)
8	9	1953	T. S. Almeida (1L) T. L. K. Mendis (2L)	D. Madugalle	Kandy	Colombo	0	0	13	3	0	16	Trinity	Trinity (16)
10	11	1955	C. V. Gunaratne	M. G. Ratwatte	Kandy	Colombo	0	0	6	0	0	6	Trinity	Trinity (16)
12	13	1957	R. Sivaratnam	K. B. de Joedt	Kandy	Colombo	0	8	8	9	8	17	Trinity	Trinity (17)
18	19	1963	M. Jayakumar	B. D. A. Piyasena	Kandy	Colombo	6	3	6	5	9	11	Trinity	Trinity (18)
22	23	1967	D. J. Perera (1L) N. B. L. Lieversz (2L)	A. L. Abeyratne	Kandy	Colombo	3	3	17	16	6	33	Trinity	Trinity (18)
24	25	1969	M. H. C. Malwatte	M. S. Jainudeen	Kandy	Colombo	0	3	8	9	3	17	Trinity	Trinity (18)
32	33	1977	S. S. Hashim	R. N. Balasuriya	Kandy	Colombo	6	4	12	10	10	22	Trinity	Trinity (20)
36	37	1981	N. S. Cooray	R. Bandaranayake	Kandy	Colombo	3	10	16	3	13	19	Trinity	Trinity (20)
38	39	1983	S. Cooray	A. D. Ratwatte	Kandy	Colombo	6	10	14	6	16	20	Trinity	Trinity (20)
40	41	1985	C. Nanayakkara	M. Jayatissa	Kandy	Colombo	9	4	18	3	13	21	Trinity	Trinity (20)
42	43	1987	H. A. S. Madanayake (1L) R. de Z. Gunarathne (2L)	T. Rajapakse	Kandy	Colombo	3	3	7	19	6	26	Trinity	Trinity (20)
46	47	1991	M. A. Deen	M. A. Deen	Kandy	Colombo	4	10	12	25	14	37	Trinity	Trinity (20)

Next steps: [Generate code with trinity_kandy_first_leg](#) [View recommended plots](#) [New interactive sheet](#)

```
import matplotlib.pyplot as plt

# Count of wins at home venues
trinity_home_wins = df[(df['1st leg Venue'].str.lower() == 'kandy') &
                      (df['Winner'].str.lower() == 'trinity')].shape[0]

royal_home_wins = df[(df['1st leg Venue'].str.lower() == 'colombo') &
                    (df['Winner'].str.lower() == 'royal')].shape[0]

# Create a simple DataFrame for plotting
home_wins = {'Team': ['Trinity', 'Royal'], 'Home Leg Wins': [trinity_home_wins, royal_home_wins]}

# Convert to pandas DataFrame
import pandas as pd
home_wins_df = pd.DataFrame(home_wins)

# Plot bar chart
plt.figure(figsize=(6,5))
bars = plt.bar(home_wins_df['Team'], home_wins_df['Home Leg Wins'], color=['red', 'blue'])

# Add labels on top
```

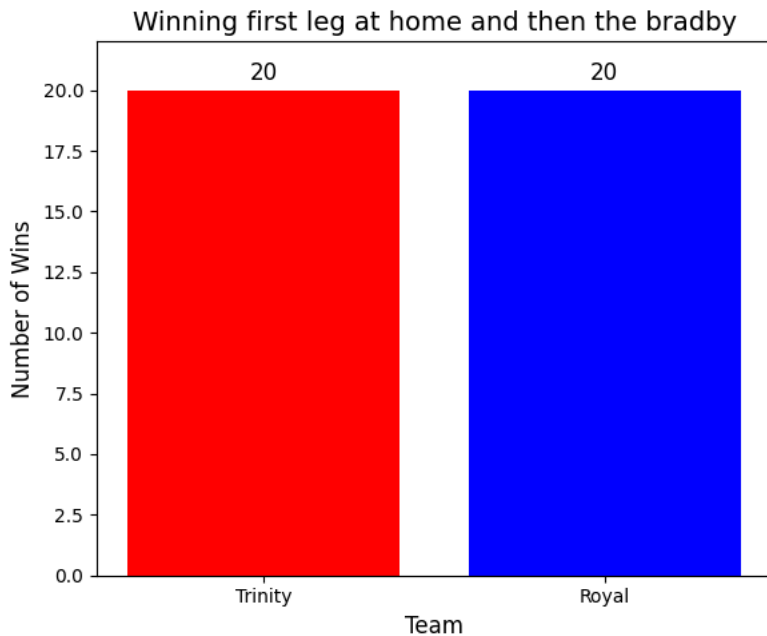


```

for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, height + 0.2, int(height), ha='center', va='bottom', fontsize=12)

plt.title('Winning first leg at home and then the bradby', fontsize=14)
plt.ylabel('Number of Wins', fontsize=12)
plt.xlabel('Team', fontsize=12)
plt.ylim(0, max(home_wins_df['Home Leg Wins']) + 2)
plt.tight_layout()
plt.show()

```



```

# Count of upset wins
trinity_upset_count = df[(df['1st leg Venue'].str.lower() == 'colombo') &
    (df['Winner'].str.lower() == 'trinity')].shape[0]

royal_upset_count = df[(df['1st leg Venue'].str.lower() == 'kandy') &
    (df['Winner'].str.lower() == 'royal')].shape[0]

# Data for plotting
upset_wins = {'Team': ['Trinity', 'Royal'],
    'Upset Wins': [trinity_upset_count, royal_upset_count]}
upset_wins_df = pd.DataFrame(upset_wins)

# Plot bar chart
plt.figure(figsize=(6,5))
bars = plt.bar(upset_wins_df['Team'], upset_wins_df['Upset Wins'], color=['red', 'blue'])

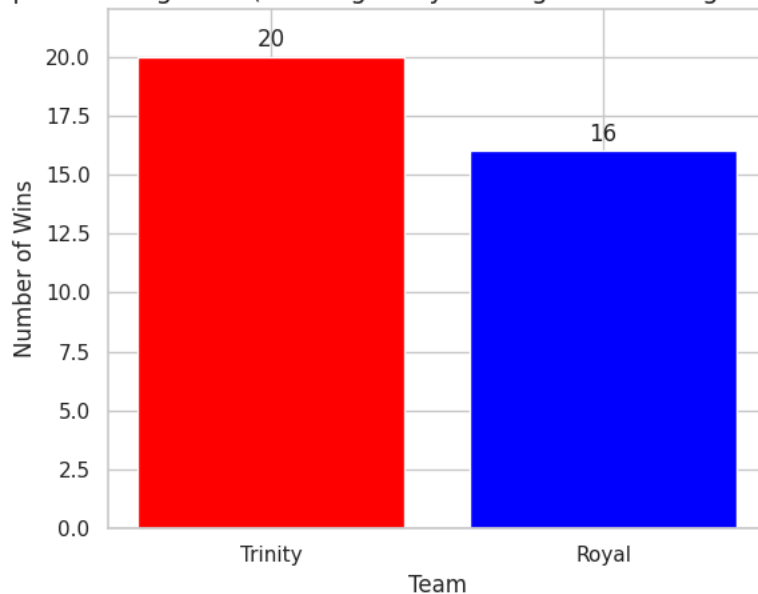
# Add labels on top of bars
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, height + 0.2, int(height),
        ha='center', va='bottom', fontsize=12)

plt.title('Upset 1st Leg Wins (Winning Away 1st Leg and Winning the Bradby)', fontsize=14)
plt.ylabel('Number of Wins', fontsize=12)
plt.xlabel('Team', fontsize=12)
plt.ylim(0, max(upset_wins_df['Upset Wins']) + 2)
plt.tight_layout()
plt.show()

```



Upset 1st Leg Wins (Winning Away 1st Leg and Winning the Bradby)



```
import pandas as pd
import matplotlib.pyplot as plt

# Get the rows for Trinity losing home leg but winning Bradby
trinity_home_comeback_rows = df[(df['1st leg Venue'].str.lower() == 'kandy') &
    (df['Royal 1st leg'] > df['Trinity 1st leg']) &
    (df['Winner'].str.lower() == 'trinity')][
    ['Year', 'Royal 1st leg', 'Trinity 1st leg', 'Royal 2nd leg', 'Trinity 2nd leg', 'Winner']
]

# Get the rows for Royal losing home leg but winning Bradby
royal_home_comeback_rows = df[(df['1st leg Venue'].str.lower() == 'colombo') &
    (df['Trinity 1st leg'] > df['Royal 1st leg']) &
    (df['Winner'].str.lower() == 'royal')][
    ['Year', 'Royal 1st leg', 'Trinity 1st leg', 'Royal 2nd leg', 'Trinity 2nd leg', 'Winner']
]

# Combine both
home_comeback_rows = pd.concat([trinity_home_comeback_rows, royal_home_comeback_rows])

# Display as a Matplotlib table
fig, ax = plt.subplots(figsize=(10, len(home_comeback_rows)*0.5 + 1))
ax.axis('tight')
ax.axis('off')
table_data = home_comeback_rows.values
column_labels = home_comeback_rows.columns
ax.table(cellText=table_data, colLabels=column_labels, cellLoc='center', loc='center',
    colColours=['white', 'lightcoral', 'lightblue', 'lightcoral', 'lightblue', 'lightgrey'])

plt.title('Losing the 1st leg at home and coming back to win the Bradby', fontsize=11)
plt.show()
```



Losing the 1st leg at home and coming back to win the Bradby

Year	Royal 1st leg	Trinity 1st leg	Royal 2nd leg	Trinity 2nd leg	Winner
2011	33	25	5	40	Trinity
1998	18	20	42	11	Royal

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# --- Count wins in Kandy for both teams ---
trinity_wins_kandy = 0
royal_wins_kandy = 0

for idx, row in df.iterrows():
```

```

# First leg
if row['1st leg Venue'].lower() == 'kandy':
    if row['Trinity 1st leg'] > row['Royal 1st leg']:
        trinity_wins_kandy += 1
    elif row['Royal 1st leg'] > row['Trinity 1st leg']:
        royal_wins_kandy += 1
# Second leg
if row['2nd leg Venue'].lower() == 'kandy':
    if row['Trinity 2nd leg'] > row['Royal 2nd leg']:
        trinity_wins_kandy += 1
    elif row['Royal 2nd leg'] > row['Trinity 2nd leg']:
        royal_wins_kandy += 1

# --- Prepare DataFrame for Seaborn ---
kandy_wins_df = pd.DataFrame({
    'Team': ['Trinity', 'Royal'],
    'Kandy Leg Wins': [trinity_wins_kandy, royal_wins_kandy]
})

# --- Plot bar chart with Seaborn ---
plt.figure(figsize=(6,6)) # taller figure for labels
sns.barplot(data=kandy_wins_df, x='Team', y='Kandy Leg Wins', palette=['red', 'blue'])

# Add numeric labels on top of bars
for index, row in kandy_wins_df.iterrows():
    plt.text(index, row['Kandy Leg Wins'] + 1, # space above bar
             int(row['Kandy Leg Wins']), ha='center', va='bottom', fontsize=12)

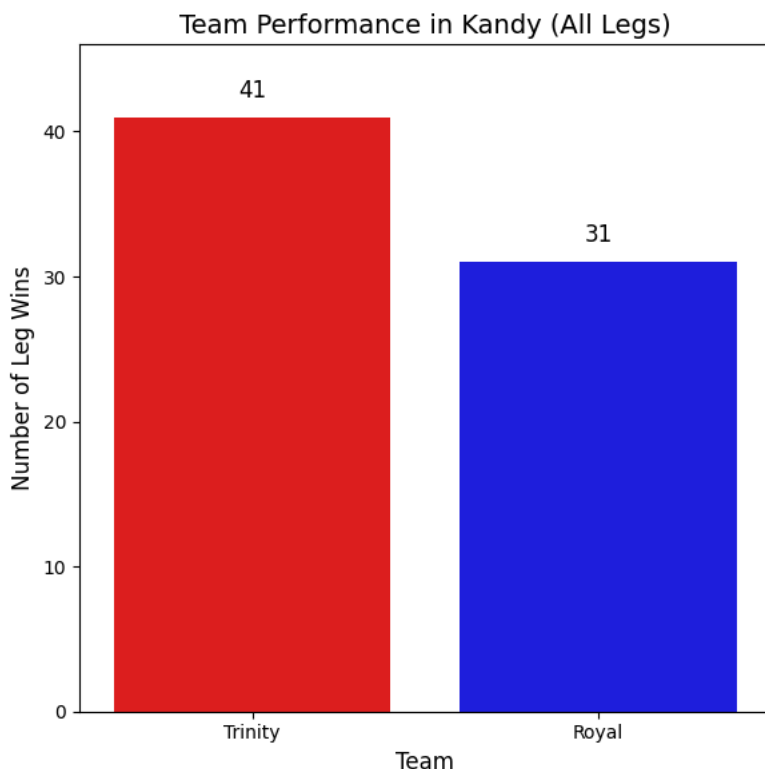
plt.title('Team Performance in Kandy (All Legs)', fontsize=14)
plt.ylabel('Number of Leg Wins', fontsize=12)
plt.xlabel('Team', fontsize=12)
plt.ylim(0, max(kandy_wins_df['Kandy Leg Wins']) + 5) # ensure tallest label fits
plt.tight_layout()
plt.show()

```

↗ /tmp/ipython-input-4100007854.py:31: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue`

```
sns.barplot(data=kandy_wins_df, x='Team', y='Kandy Leg Wins', palette=['red', 'blue'])
```



```

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

```

```

# --- Count wins in Colombo for both teams ---
trinity_wins_colombo = 0
royal_wins_colombo = 0

```

```

for idx, row in df.iterrows():
    # First leg
    if row['1st leg Venue'].lower() == 'colombo':
        if row['Trinity 1st leg'] > row['Royal 1st leg']:
            trinity_wins_colombo += 1
        elif row['Royal 1st leg'] > row['Trinity 1st leg']:
            royal_wins_colombo += 1
    # Second leg
    if row['2nd leg Venue'].lower() == 'colombo':
        if row['Trinity 2nd leg'] > row['Royal 2nd leg']:
            trinity_wins_colombo += 1
        elif row['Royal 2nd leg'] > row['Trinity 2nd leg']:
            royal_wins_colombo += 1

# --- Prepare DataFrame for Seaborn ---
colombo_wins_df = pd.DataFrame({
    'Team': ['Trinity', 'Royal'],
    'Colombo Leg Wins': [trinity_wins_colombo, royal_wins_colombo]
})

# --- Plot bar chart with Seaborn ---
plt.figure(figsize=(6,6)) # taller figure for labels
sns.barplot(data=colombo_wins_df, x='Team', y='Colombo Leg Wins', palette=['red', 'blue'])

# Add numeric labels on top of bars
for index, row in colombo_wins_df.iterrows():
    plt.text(index, row['Colombo Leg Wins'] + 1, # space above bar
             int(row['Colombo Leg Wins']), ha='center', va='bottom', fontsize=12)

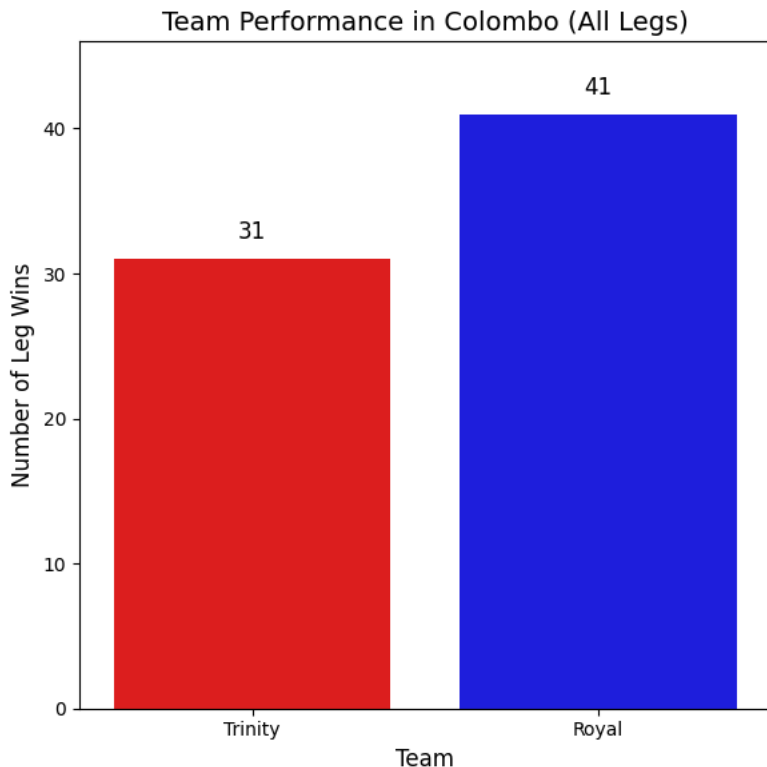
plt.title('Team Performance in Colombo (All Legs)', fontsize=14)
plt.ylabel('Number of Leg Wins', fontsize=12)
plt.xlabel('Team', fontsize=12)
plt.ylim(0, max(colombo_wins_df['Colombo Leg Wins']) + 5) # ensure tallest label fits
plt.tight_layout()
plt.show()

```

↗ /tmp/ipython-input-3300512974.py:32: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue`

```
sns.barplot(data=colombo_wins_df, x='Team', y='Colombo Leg Wins', palette=['red', 'blue'])
```



```

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# --- Initialize counters ---
trinity_first_leg_kandy = 0

```

```

royal_first_leg_kandy = 0

# --- Count first-leg wins in Kandy ---
for idx, row in df.iterrows():
    if row['1st leg Venue'].lower() == 'kandy':
        if row['Trinity 1st leg'] > row['Royal 1st leg']:
            trinity_first_leg_kandy += 1
        elif row['Royal 1st leg'] > row['Trinity 1st leg']:
            royal_first_leg_kandy += 1

# --- Prepare DataFrame for plotting ---
kandy_first_leg_wins_df = pd.DataFrame({
    'Team': ['Trinity', 'Royal'],
    'First-Leg Wins in Kandy': [trinity_first_leg_kandy, royal_first_leg_kandy]
})

# --- Plot bar chart ---
plt.figure(figsize=(6,6))
sns.barplot(data=kandy_first_leg_wins_df, x='Team', y='First-Leg Wins in Kandy', palette=['red', 'blue'])

# Add numeric labels on top of bars
for index, row in kandy_first_leg_wins_df.iterrows():
    plt.text(index, row['First-Leg Wins in Kandy'] + 0.3, int(row['First-Leg Wins in Kandy']),
             ha='center', va='bottom', fontsize=12)

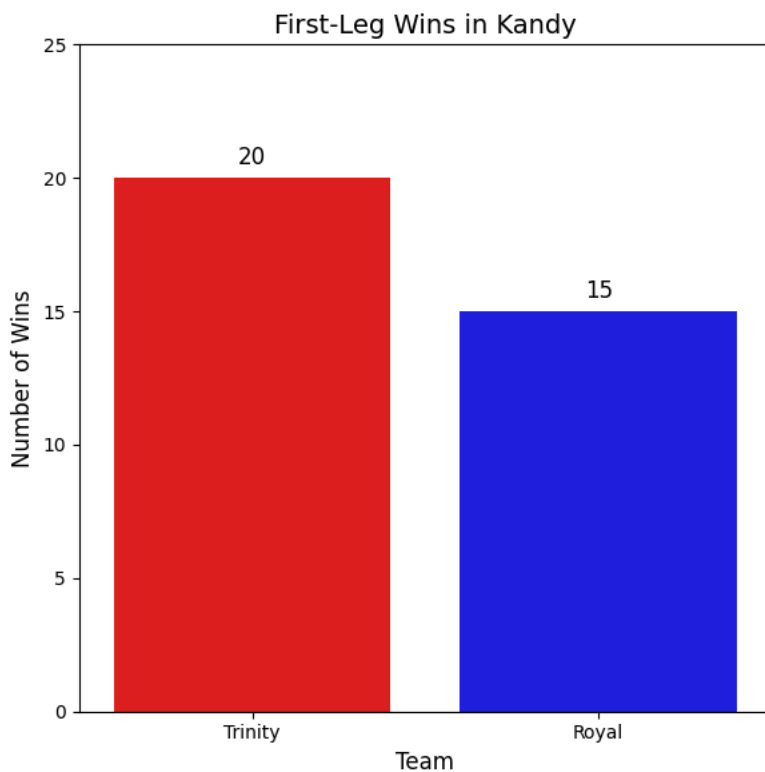
plt.title('First-Leg Wins in Kandy', fontsize=14)
plt.ylabel('Number of Wins', fontsize=12)
plt.xlabel('Team', fontsize=12)
plt.ylim(0, max(kandy_first_leg_wins_df['First-Leg Wins in Kandy']) + 5)
plt.tight_layout()
plt.show()

```

↗ /tmp/ipython-input-3461557539.py:25: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue`

```
sns.barplot(data=kandy_first_leg_wins_df, x='Team', y='First-Leg Wins in Kandy', palette=['red', 'blue'])
```



```

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

```

```

# --- Initialize counters ---
trinity_first_leg_colombo = 0
royal_first_leg_colombo = 0

```

```

# --- Count first-leg wins in Colombo ---
for idx, row in df.iterrows():
    if row['1st leg Venue'].lower() == 'colombo':

```

```

    if row['Trinity 1st leg'] > row['Royal 1st leg']:
        trinity_first_leg_colombo += 1
    elif row['Royal 1st leg'] > row['Trinity 1st leg']:
        royal_first_leg_colombo += 1

# --- Prepare DataFrame for plotting ---
colombo_first_leg_wins_df = pd.DataFrame({
    'Team': ['Trinity', 'Royal'],
    'First-Leg Wins in Colombo': [trinity_first_leg_colombo, royal_first_leg_colombo]
})

# --- Plot bar chart ---
plt.figure(figsize=(6,6))
sns.barplot(data=colombo_first_leg_wins_df, x='Team', y='First-Leg Wins in Colombo', palette=['red', 'blue'])

# Add numeric labels on top of bars
for index, row in colombo_first_leg_wins_df.iterrows():
    plt.text(index, row['First-Leg Wins in Colombo'] + 0.3, int(row['First-Leg Wins in Colombo']),
             ha='center', va='bottom', fontsize=12)

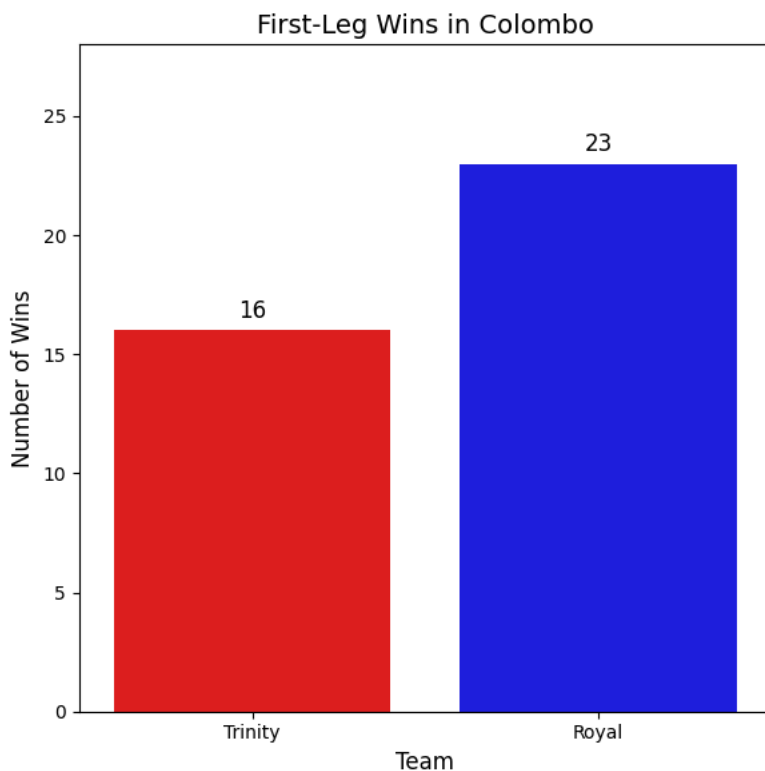
plt.title('First-Leg Wins in Colombo', fontsize=14)
plt.ylabel('Number of Wins', fontsize=12)
plt.xlabel('Team', fontsize=12)
plt.ylim(0, max(colombo_first_leg_wins_df['First-Leg Wins in Colombo']) + 5)
plt.tight_layout()
plt.show()

```

↗ /tmp/ipython-input-4189131513.py:25: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue`

```
sns.barplot(data=colombo_first_leg_wins_df, x='Team', y='First-Leg Wins in Colombo', palette=['red', 'blue'])
```



```

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# --- Initialize counters ---
trinity_second_leg_colombo = 0
royal_second_leg_colombo = 0

# --- Count second-leg wins in Colombo ---
for idx, row in df.iterrows():
    if row['2nd leg Venue'].lower() == 'colombo':
        if row['Trinity 2nd leg'] > row['Royal 2nd leg']:
            trinity_second_leg_colombo += 1
        elif row['Royal 2nd leg'] > row['Trinity 2nd leg']:
            royal_second_leg_colombo += 1

```

```
# --- Prepare DataFrame for plotting ---
colombo_second_leg_wins_df = pd.DataFrame({
    'Team': ['Trinity', 'Royal'],
    'Second-Leg Wins in Colombo': [trinity_second_leg_colombo, royal_second_leg_colombo]
})

# --- Plot bar chart ---
plt.figure(figsize=(6,6))
sns.barplot(data=colombo_second_leg_wins_df, x='Team', y='Second-Leg Wins in Colombo', palette=['red', 'blue'])

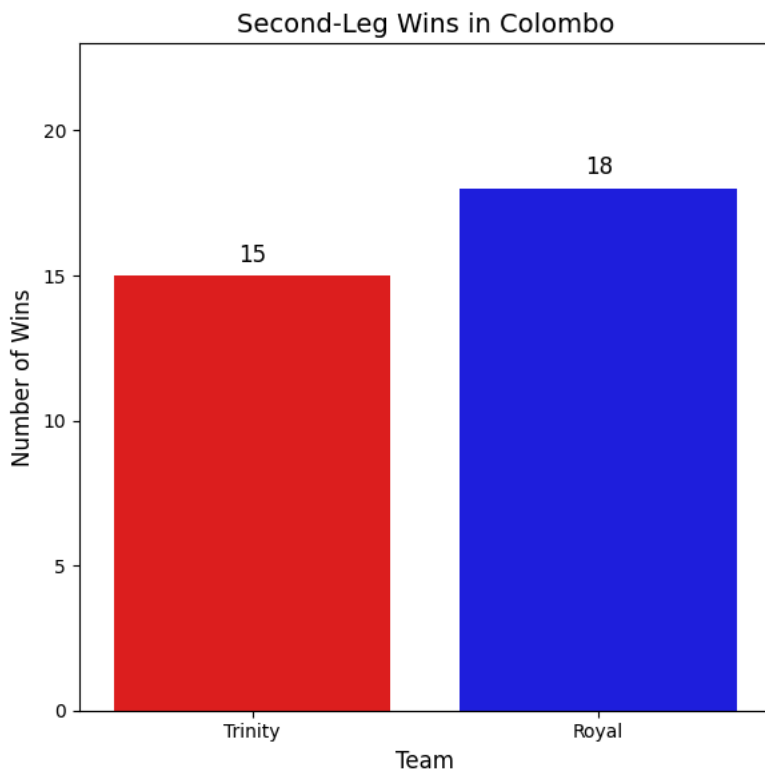
# Add numeric labels on top of bars
for index, row in colombo_second_leg_wins_df.iterrows():
    plt.text(index, row['Second-Leg Wins in Colombo'] + 0.3, int(row['Second-Leg Wins in Colombo']),
             ha='center', va='bottom', fontsize=12)

plt.title('Second-Leg Wins in Colombo', fontsize=14)
plt.ylabel('Number of Wins', fontsize=12)
plt.xlabel('Team', fontsize=12)
plt.ylim(0, max(colombo_second_leg_wins_df['Second-Leg Wins in Colombo']) + 5)
plt.tight_layout()
plt.show()
```

→ /tmp/ipython-input-836411946.py:25: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue`

```
sns.barplot(data=colombo_second_leg_wins_df, x='Team', y='Second-Leg Wins in Colombo', palette=['red', 'blue'])
```



```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# --- Initialize counters ---
trinity_second_leg_kandy = 0
royal_second_leg_kandy = 0

# --- Count second-leg wins in Kandy ---
for idx, row in df.iterrows():
    if row['2nd leg Venue'].lower() == 'kandy':
        if row['Trinity 2nd leg'] > row['Royal 2nd leg']:
            trinity_second_leg_kandy += 1
        elif row['Royal 2nd leg'] > row['Trinity 2nd leg']:
            royal_second_leg_kandy += 1

# --- Prepare DataFrame for plotting ---
kandy_second_leg_wins_df = pd.DataFrame({
    'Team': ['Trinity', 'Royal'],
    'Second-Leg Wins in Kandy': [trinity_second_leg_kandy, royal_second_leg_kandy]
})
```

```
# --- Plot bar chart ---
plt.figure(figsize=(6,6))
sns.barplot(data=kandy_second_leg_wins_df, x='Team', y='Second-Leg Wins in Kandy', palette=['red', 'blue'])

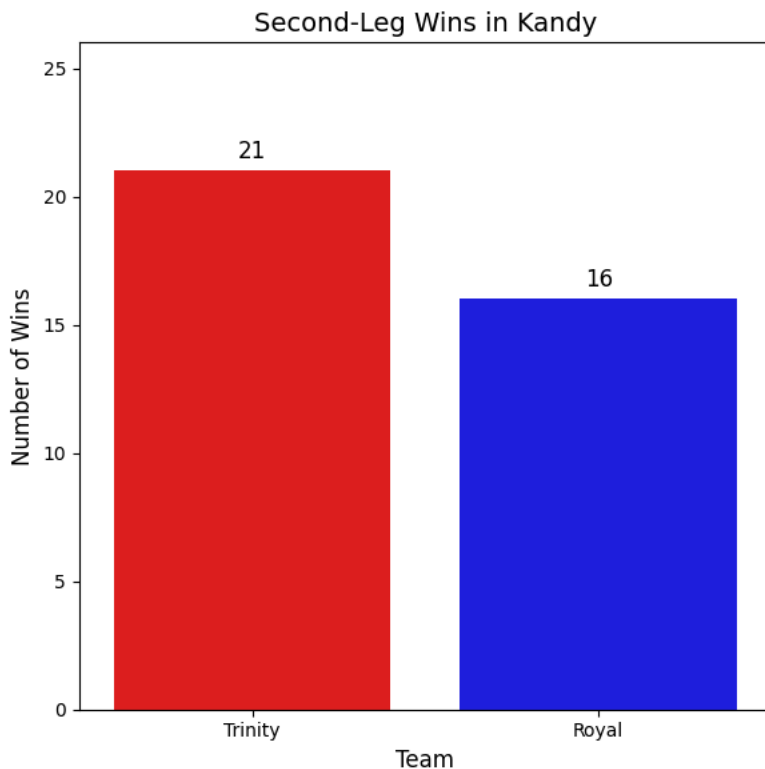
# Add numeric labels on top of bars
for index, row in kandy_second_leg_wins_df.iterrows():
    plt.text(index, row['Second-Leg Wins in Kandy'] + 0.3, int(row['Second-Leg Wins in Kandy']),
             ha='center', va='bottom', fontsize=12)

plt.title('Second-Leg Wins in Kandy', fontsize=14)
plt.ylabel('Number of Wins', fontsize=12)
plt.xlabel('Team', fontsize=12)
plt.ylim(0, max(kandy_second_leg_wins_df['Second-Leg Wins in Kandy']) + 5)
plt.tight_layout()
plt.show()
```

↗ /tmp/ipython-input-424434805.py:25: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue`

```
sns.barplot(data=kandy_second_leg_wins_df, x='Team', y='Second-Leg Wins in Kandy', palette=['red', 'blue'])
```



```
import pandas as pd

# Filter rows where first leg venue is Kandy
kandy_first_leg = df[df['1st leg Venue'].str.lower() == 'kandy']

# Count winners (Trinity, Royal) and Draws
winner_counts = kandy_first_leg['Winner'].value_counts()

# Ensure all categories are included
for team in ['Trinity', 'Royal', 'Draw']:
    if team not in winner_counts:
        winner_counts[team] = 0

# Reorder for clarity
winner_counts = winner_counts[['Trinity', 'Royal', 'Draw']]

print("Bradby Winners when First Leg is in Kandy:")
print(winner_counts)
```

↗ Bradby Winners when First Leg is in Kandy:

Winner	
Trinity	20
Royal	16
Draw	0

Name: count, dtype: int64


```
# Convert to DataFrame for Seaborn
winner_counts_df = winner_counts.reset_index()
winner_counts_df.columns = ['Team', 'Bradby Wins']

# --- Plot using Seaborn ---
plt.figure(figsize=(6,6))
sns.barplot(data=winner_counts_df, x='Team', y='Bradby Wins',
            palette=['red', 'blue', 'green']) # red=Trinity, blue=Royal, green=Draw

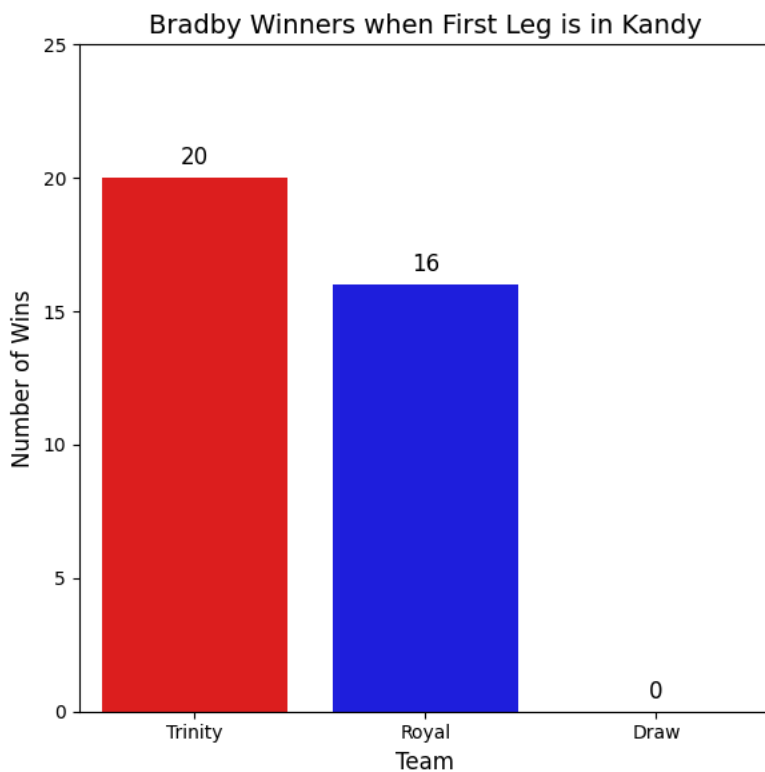
# Add numeric labels on top of bars
for idx, row in winner_counts_df.iterrows():
    plt.text(idx, row['Bradby Wins'] + 0.3, int(row['Bradby Wins']),
            ha='center', va='bottom', fontsize=12)

plt.title('Bradby Winners when First Leg is in Kandy', fontsize=14)
plt.ylabel('Number of Wins', fontsize=12)
plt.xlabel('Team', fontsize=12)
plt.ylim(0, max(winner_counts_df['Bradby Wins']) + 5)
plt.tight_layout()
plt.show()
```

↗ /tmp/ipython-input-3104602190.py:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue`

```
sns.barplot(data=winner_counts_df, x='Team', y='Bradby Wins',
```



```
import pandas as pd

# Filter rows where first leg venue is Colombo
colombo_first_leg = df[df['1st leg Venue'].str.lower() == 'colombo']

# Count winners (Trinity, Royal) and Draws
winner_counts_colombo = colombo_first_leg['Winner'].value_counts()

# Ensure all categories are included
for team in ['Trinity', 'Royal', 'Draw']:
    if team not in winner_counts_colombo:
        winner_counts_colombo[team] = 0

# Reorder for clarity
winner_counts_colombo = winner_counts_colombo[['Trinity', 'Royal', 'Draw']]

print("Bradby Winners when First Leg is in Colombo:")
print(winner_counts_colombo)
```

↗ Bradby Winners when First Leg is in Colombo:
Winner

```

Trinity    20
Royal      20
Draw        2
Name: count, dtype: int64

```

```

import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Convert to DataFrame for plotting
winner_counts_colombo_df = winner_counts_colombo.reset_index()
winner_counts_colombo_df.columns = ['Team', 'Bradby Wins']

# --- Plot using Seaborn ---
plt.figure(figsize=(6,6))
sns.barplot(data=winner_counts_colombo_df, x='Team', y='Bradby Wins',
            palette=['red', 'blue', 'green']) # red=Trinity, blue=Royal, green=Draw

# Add numeric labels on top of bars
for idx, row in winner_counts_colombo_df.iterrows():
    plt.text(idx, row['Bradby Wins'] + 0.3, int(row['Bradby Wins']),
             ha='center', va='bottom', fontsize=12)

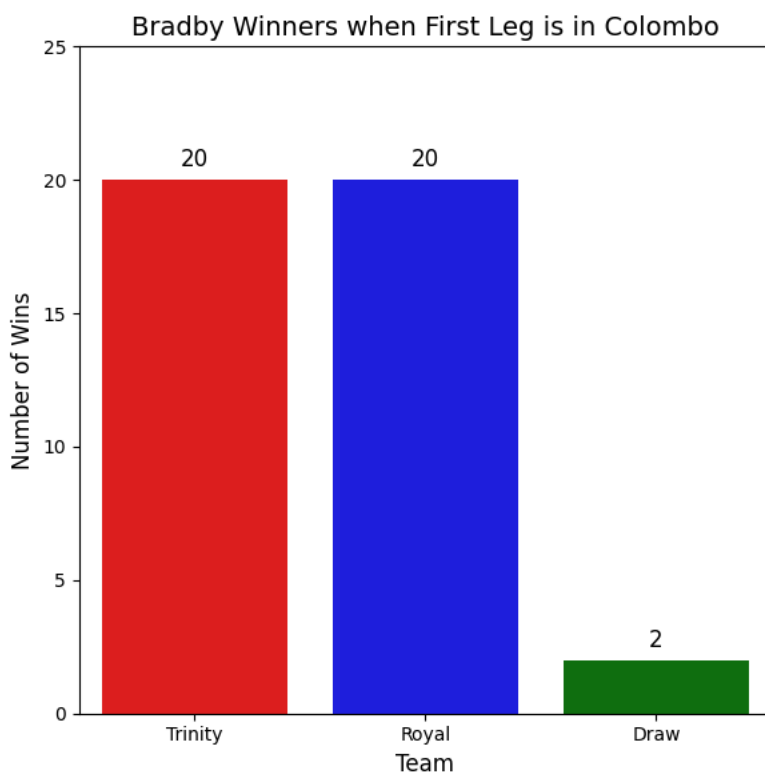
plt.title('Bradby Winners when First Leg is in Colombo', fontsize=14)
plt.ylabel('Number of Wins', fontsize=12)
plt.xlabel('Team', fontsize=12)
plt.ylim(0, max(winner_counts_colombo_df['Bradby Wins']) + 5)
plt.tight_layout()
plt.show()

```

↗ /tmp/ipython-input-1826540535.py:11: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue`

```
sns.barplot(data=winner_counts_colombo_df, x='Team', y='Bradby Wins',
```



✓ Over the Years

```

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Map winners to numeric codes for heatmap
winner_mapping = {'Trinity': 0, 'Royal': 1, 'Draw': 2}
df['Winner_Code'] = df['Winner'].map(winner_mapping)

# Sort by year

```

```

df_sorted = df.sort_values('Year')

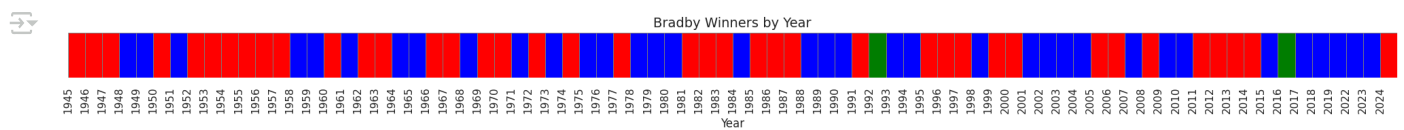
# Create a DataFrame with one row, columns = years
heatmap_data = pd.DataFrame([df_sorted['Winner_Code'].values],
                             columns=df_sorted['Year'])

# Make figure taller to increase tile height
plt.figure(figsize=(20, 2)) # Wider figure for readability

sns.heatmap(
    heatmap_data,
    cmap=['red', 'blue', 'green'], # Trinity, Royal, Draw
    cbar=False,
    linewidths=0.5,
    linecolor='gray',
    square=False, # Important: allows vertical stretching
)

# Show all years as x-ticks
plt.xticks(ticks=range(len(df_sorted['Year'])), labels=df_sorted['Year'], rotation=90)
plt.yticks([]) # Hide y-axis
plt.title('Bradby Winners by Year', fontsize=14)
plt.tight_layout()
plt.show()

```



```

import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Filter out Draw
df_no_draw = df[df['Winner'].isin(['Trinity', 'Royal'])].copy()

# Sort by Year
df_sorted = df_no_draw.sort_values('Year')

# Calculate cumulative wins
cum_wins = df_sorted.groupby('Winner')['Year'].value_counts().unstack(fill_value=0).cumsum(axis=1).T

plt.figure(figsize=(15,5))
sns.lineplot(data=cum_wins, palette={'Trinity':'red','Royal':'blue'}, linewidth=2.5, marker='o')

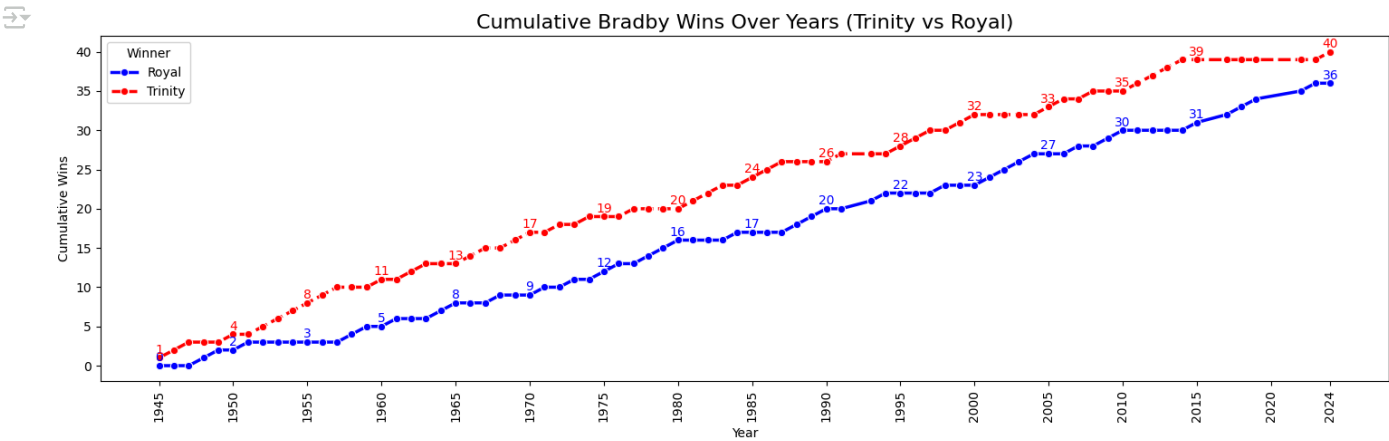
plt.title('Cumulative Bradby Wins Over Years (Trinity vs Royal)', fontsize=16)
plt.xlabel('Year')
plt.ylabel('Cumulative Wins')

# X-axis: every 5 years from first year
start_year = cum_wins.index.min()
end_year = cum_wins.index.max()
five_year_ticks = np.arange(start_year, end_year + 1, 5)
# Always include 2024 even if not on 5-year boundary
if 2024 not in five_year_ticks:
    five_year_ticks = np.append(five_year_ticks, 2024)
plt.xticks(ticks=five_year_ticks, labels=five_year_ticks, rotation=90)

# Add cumulative labels every 5 years + 2024
for year in five_year_ticks:
    if year in cum_wins.index:
        for team in cum_wins.columns:
            y = cum_wins.loc[year, team]
            plt.text(year, y + 0.2, int(y), color='red' if team=='Trinity' else 'blue',
                     fontsize=10, ha='center', va='bottom')

plt.tight_layout()
plt.show()

```



```
# Filter out draws
df_filtered = df[df['Winner'].isin(['Trinity', 'Royal'])].copy()

# Sort by Year
df_sorted = df_filtered.sort_values('Year')

# Create a table: years as index, teams as columns, counting wins per year
win_counts = df_sorted.groupby('Winner')['Year'].value_counts().unstack(fill_value=0)

# Compute cumulative wins over the years
cum_wins = win_counts.cumsum(axis=1).T # transpose so years are rows
cum_wins.index.name = 'Year'
cum_wins
```

Winner	Royal	Trinity
Year		
1945	0	1
1946	0	2
1947	0	3
1948	1	3
1949	2	3
...
2018	33	39
2019	34	39
2022	35	39
2023	36	39
2024	36	40

76 rows x 2 columns

Next steps:

Generate code with cum_wins

☒ View recommended plots

New interactive sheet

```
# Make sure only Trinity and Royal columns exist
cum_wins_subset = cum_wins[['Trinity', 'Royal']].copy()

# Calculate the absolute gap
cum_wins_subset['Gap'] = abs(cum_wins_subset['Trinity'] - cum_wins_subset['Royal'])

# Sort by gap descending and get top 5
top5_gaps = cum_wins_subset.sort_values(by='Gap', ascending=False).head(10)

# Optionally, add a column showing who is ahead
top5_gaps['Leader'] = top5_gaps.apply(lambda row: 'Trinity' if row['Trinity'] > row['Royal'] else 'Royal', axis=1)

top5_gaps
```

Winner	Trinity	Royal	Gap	Leader
Year				
2000	32	23	9	Trinity
2014	39	30	9	Trinity
1987	26	17	9	Trinity
2013	38	30	8	Trinity
1972	18	10	8	Trinity
1974	19	11	8	Trinity
1986	25	17	8	Trinity
1988	26	18	8	Trinity
1970	17	9	8	Trinity
2015	39	31	8	Trinity

Next steps:

[Generate code with top5_gaps](#)[View recommended plots](#)[New interactive sheet](#)

```
# Make sure only Trinity and Royal columns exist
cum_wins_subset = cum_wins[['Trinity', 'Royal']].copy()

# Calculate the absolute gap
cum_wins_subset['Gap'] = abs(cum_wins_subset['Trinity'] - cum_wins_subset['Royal'])

# Sort by gap ascending and get top 5 (smallest gaps)
smallest5_gaps = cum_wins_subset.sort_values(by='Gap', ascending=True).head(10)

# Optionally, add a column showing who is ahead
smallest5_gaps['Leader'] = smallest5_gaps.apply(lambda row: 'Trinity' if row['Trinity'] > row['Royal'] else ('Royal' if row['Royal'] > row['Trinity'] else 'Tie'), axis=1)

smallest5_gaps
```

Winner	Trinity	Royal	Gap	Leader
Year				
1945	1	0	1	Trinity
1949	3	2	1	Trinity
1951	4	3	1	Trinity
1946	2	0	2	Trinity
1950	4	2	2	Trinity
1948	3	1	2	Trinity
1952	5	3	2	Trinity
1947	3	0	3	Trinity
1953	6	3	3	Trinity
2023	39	36	3	Trinity

Next steps:

[Generate code with smallest5_gaps](#)[View recommended plots](#)[New interactive sheet](#)

```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Prepare data (works whether Year is index or a column)
plot_df = cum_wins_subset.reset_index() if cum_wins_subset.index.name is not None else cum_wins_subset.copy()
if 'Year' not in plot_df.columns:
    plot_df.rename(columns={plot_df.columns[0]: 'Year'}, inplace=True)

# Ensure Gap exists
# plot_df['Gap'] = (plot_df['Trinity'] - plot_df['Royal']).abs() # uncomment if needed

# Build y-axis so 9 is included
ymax = int(np.ceil(max(9, plot_df['Gap'].max())))
yticks = np.arange(0, ymax + 1, 1)

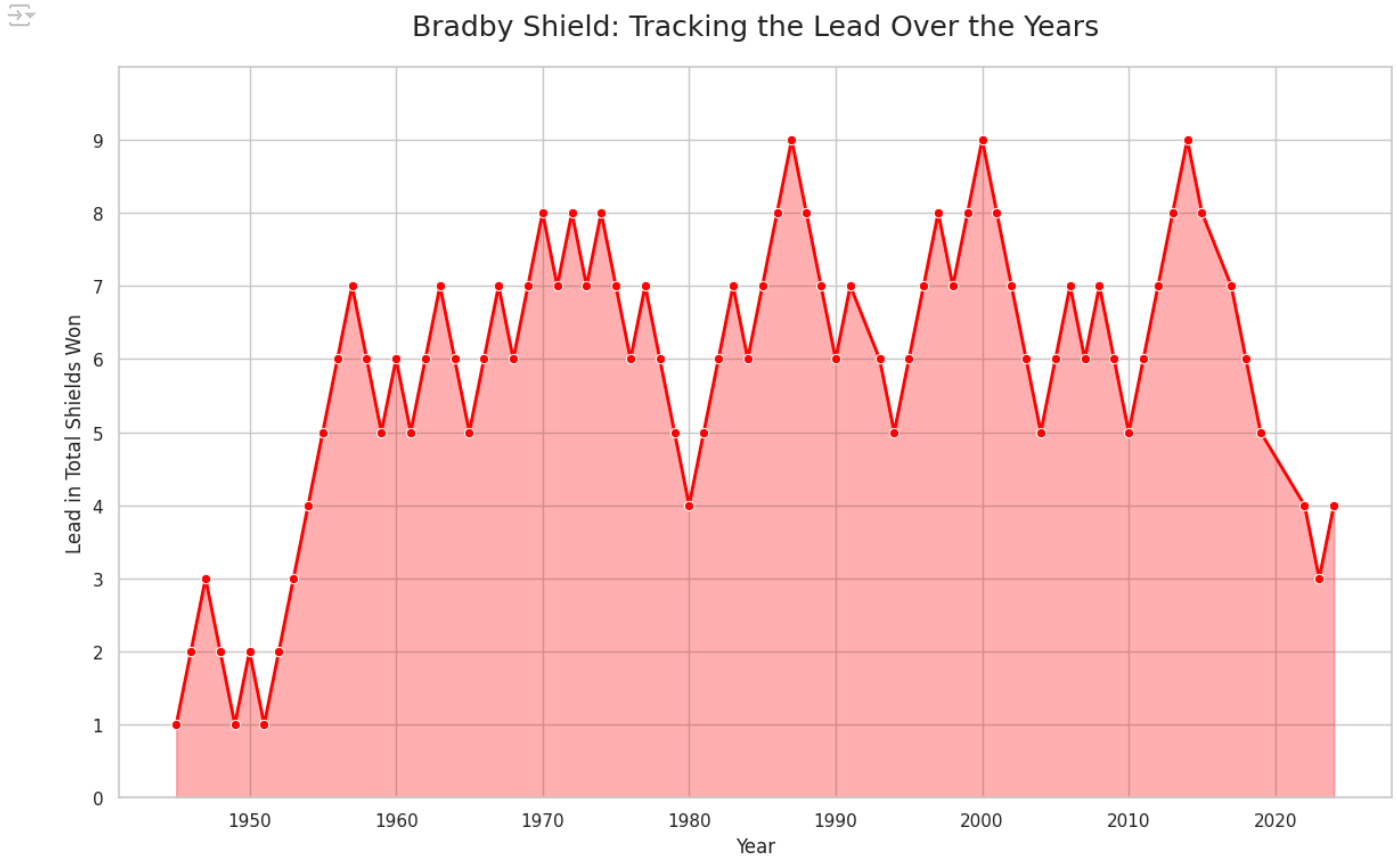
sns.set_theme(style="whitegrid")

plt.figure(figsize=(12,8)) # 🖱️ taller chart (height=8 instead of 6)
```

```
plt.fill_between(plot_df['Year'], plot_df['Gap'], color="red", alpha=0.3)
sns.lineplot(x="Year", y="Gap", data=plot_df, color="red", linewidth=2, marker="o")

plt.ylim(0, ymax + 1) # 🐞 add extra headroom
plt.yticks(yticks)

plt.title("Bradby Shield: Tracking the Lead Over the Years", fontsize=18, pad=20) # 🐞 extra padding
plt.xlabel("Year", fontsize=12)
plt.ylabel("Lead in Total Shields Won", fontsize=12)
plt.tight_layout(rect=[0, 0, 1, 0.97]) # 🐞 leaves space for title
plt.show()
```



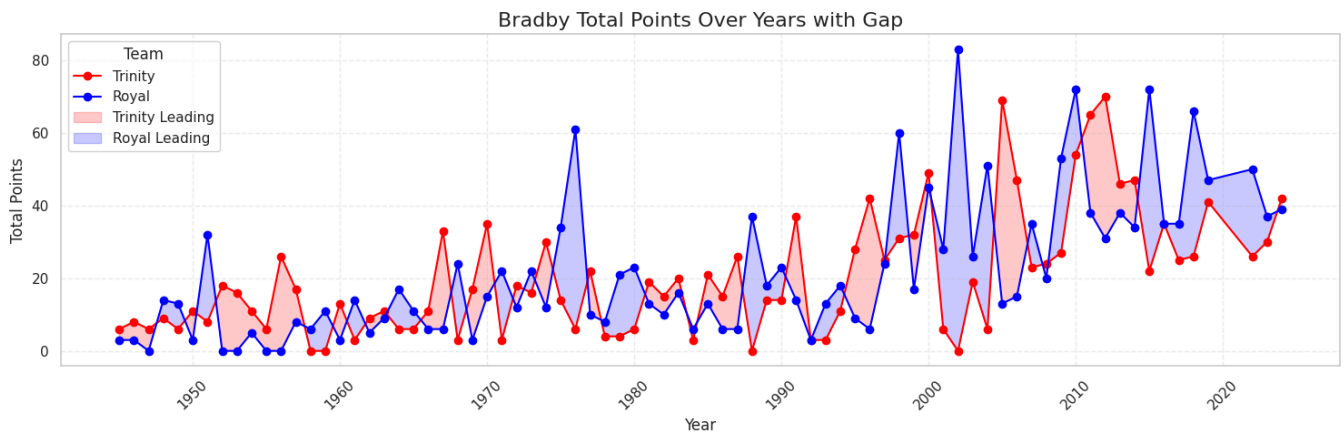
```
plt.figure(figsize=(15,5))

# Plot lines first
plt.plot(df['Year'], df['Trinity Total'], color='red', marker='o', label='Trinity')
plt.plot(df['Year'], df['Royal Total'], color='blue', marker='o', label='Royal')

# Shade area between the lines
plt.fill_between(df['Year'], df['Trinity Total'], df['Royal Total'],
                 where=(df['Trinity Total'] >= df['Royal Total']),
                 interpolate=True, color='red', alpha=0.2, label='Trinity Leading')

plt.fill_between(df['Year'], df['Trinity Total'], df['Royal Total'],
                 where=(df['Royal Total'] > df['Trinity Total']),
                 interpolate=True, color='blue', alpha=0.2, label='Royal Leading')

plt.title('Bradby Total Points Over Years with Gap', fontsize=16)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Total Points', fontsize=12)
plt.xticks(rotation=45)
plt.grid(True, linestyle='--', alpha=0.3)
plt.legend(title='Team')
plt.tight_layout()
plt.show()
```



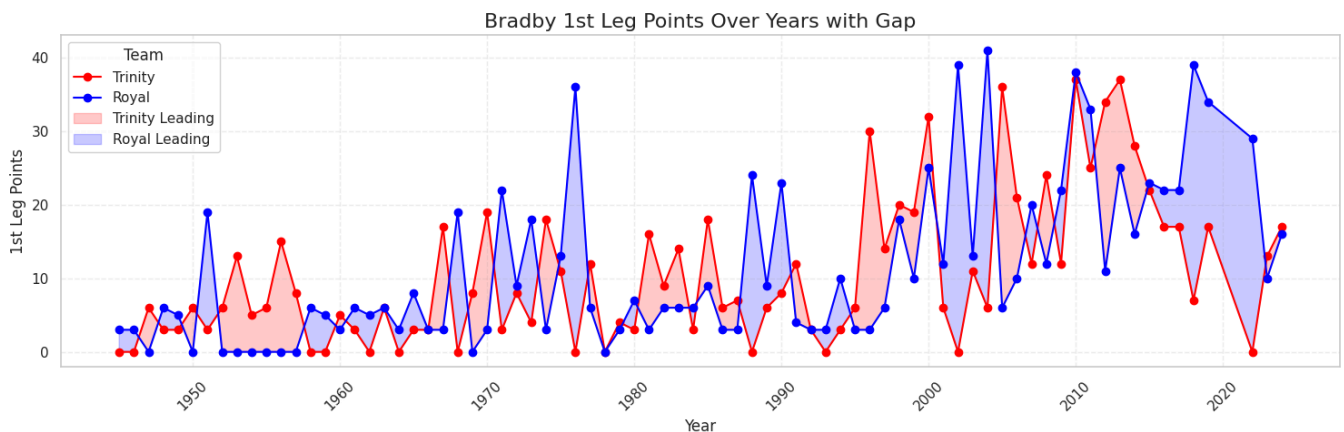
```
plt.figure(figsize=(15,5))

# Plot lines first for 1st leg scores
plt.plot(df['Year'], df['Trinity 1st leg'], color='red', marker='o', label='Trinity')
plt.plot(df['Year'], df['Royal 1st leg'], color='blue', marker='o', label='Royal')

# Shade area between the lines to show which team was leading
plt.fill_between(df['Year'], df['Trinity 1st leg'], df['Royal 1st leg'],
                 where=(df['Trinity 1st leg'] >= df['Royal 1st leg']),
                 interpolate=True, color='red', alpha=0.2, label='Trinity Leading')

plt.fill_between(df['Year'], df['Trinity 1st leg'], df['Royal 1st leg'],
                 where=(df['Royal 1st leg'] > df['Trinity 1st leg']),
                 interpolate=True, color='blue', alpha=0.2, label='Royal Leading')

plt.title('Bradby 1st Leg Points Over Years with Gap', fontsize=16)
plt.xlabel('Year', fontsize=12)
plt.ylabel('1st Leg Points', fontsize=12)
plt.xticks(rotation=45)
plt.grid(True, linestyle='--', alpha=0.3)
plt.legend(title='Team')
plt.tight_layout()
plt.show()
```



```
plt.figure(figsize=(15,5))

# Plot lines first for 2nd leg scores
plt.plot(df['Year'], df['Trinity 2nd leg'], color='red', marker='o', label='Trinity')
plt.plot(df['Year'], df['Royal 2nd leg'], color='blue', marker='o', label='Royal')

# Shade area between the lines to show which team was leading
plt.fill_between(df['Year'], df['Trinity 2nd leg'], df['Royal 2nd leg'],
                 where=(df['Trinity 2nd leg'] >= df['Royal 2nd leg']),
                 interpolate=True, color='red', alpha=0.2, label='Trinity Leading')
plt.fill_between(df['Year'], df['Trinity 2nd leg'], df['Royal 2nd leg'],
                 where=(df['Royal 2nd leg'] > df['Trinity 2nd leg']),
                 interpolate=True, color='blue', alpha=0.2, label='Royal Leading')
```

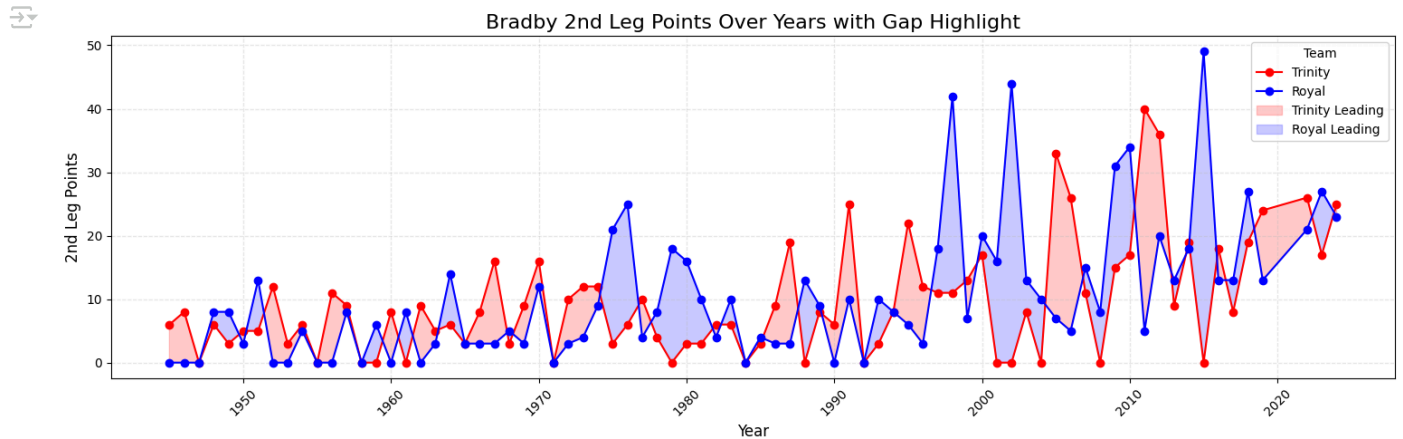
```

interpolate=True, color='red', alpha=0.2, label='Trinity Leading')

plt.fill_between(df['Year'], df['Trinity 2nd leg'], df['Royal 2nd leg'],
                 where=(df['Royal 2nd leg'] > df['Trinity 2nd leg']),
                 interpolate=True, color='blue', alpha=0.2, label='Royal Leading')

plt.title('Bradby 2nd Leg Points Over Years with Gap Highlight', fontsize=16)
plt.xlabel('Year', fontsize=12)
plt.ylabel('2nd Leg Points', fontsize=12)
plt.xticks(rotation=45)
plt.grid(True, linestyle='--', alpha=0.3)
plt.legend(title='Team')
plt.tight_layout()
plt.show()

```



```

import matplotlib.pyplot as plt

# Prepare data for plotting
kandy_years = []
trinity_scores = []
royal_scores = []

for _, row in df.iterrows():
    if row['1st leg Venue'].lower() == 'kandy':
        kandy_years.append(row['Year'])
        trinity_scores.append(row['Trinity 1st leg'])
        royal_scores.append(row['Royal 1st leg'])
    elif row['2nd leg Venue'].lower() == 'kandy':
        kandy_years.append(row['Year'])
        trinity_scores.append(row['Trinity 2nd leg'])
        royal_scores.append(row['Royal 2nd leg'])

plt.figure(figsize=(15,5))

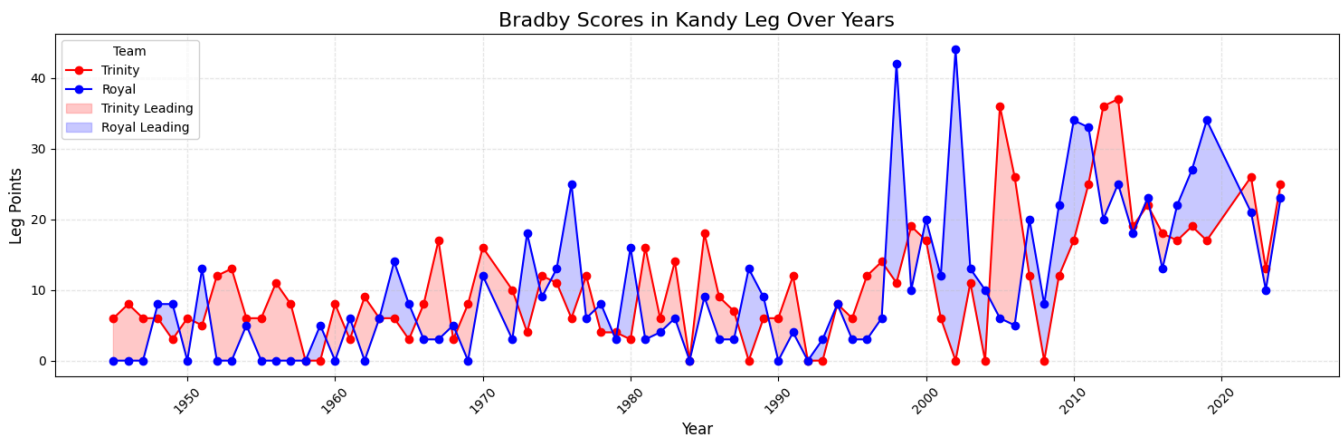
# Plot lines
plt.plot(kandy_years, trinity_scores, color='red', marker='o', label='Trinity')
plt.plot(kandy_years, royal_scores, color='blue', marker='o', label='Royal')

# Shade area to show leading team
plt.fill_between(kandy_years, trinity_scores, royal_scores,
                 where=[t >= r for t, r in zip(trinity_scores, royal_scores)],
                 interpolate=True, color='red', alpha=0.2, label='Trinity Leading')

plt.fill_between(kandy_years, trinity_scores, royal_scores,
                 where=[r > t for t, r in zip(trinity_scores, royal_scores)],
                 interpolate=True, color='blue', alpha=0.2, label='Royal Leading')

plt.title('Bradby Scores in Kandy Leg Over Years', fontsize=16)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Leg Points', fontsize=12)
plt.xticks(rotation=45)
plt.grid(True, linestyle='--', alpha=0.3)
plt.legend(title='Team')
plt.tight_layout()
plt.show()

```

```
import matplotlib.pyplot as plt

# Prepare data for plotting
colombo_years = []
trinity_scores = []
royal_scores = []

for _, row in df.iterrows():
    if row['1st leg Venue'].lower() == 'colombo':
        colombo_years.append(row['Year'])
        trinity_scores.append(row['Trinity 1st leg'])
        royal_scores.append(row['Royal 1st leg'])
    elif row['2nd leg Venue'].lower() == 'colombo':
        colombo_years.append(row['Year'])
        trinity_scores.append(row['Trinity 2nd leg'])
        royal_scores.append(row['Royal 2nd leg'])

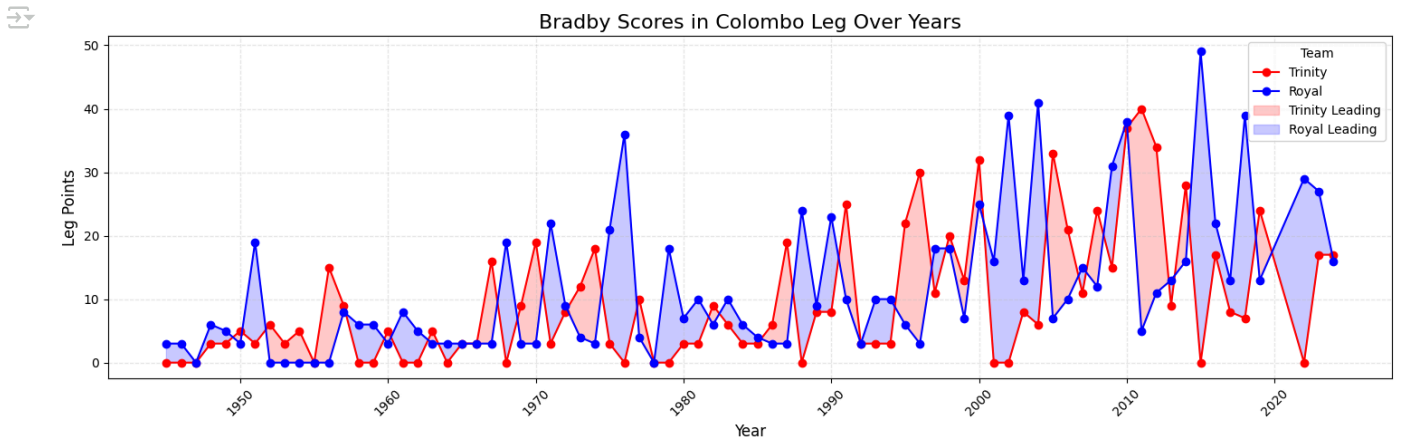
plt.figure(figsize=(15,5))

# Plot lines
plt.plot(colombo_years, trinity_scores, color='red', marker='o', label='Trinity')
plt.plot(colombo_years, royal_scores, color='blue', marker='o', label='Royal')

# Shade area to show leading team
plt.fill_between(colombo_years, trinity_scores, royal_scores,
                 where=[t >= r for t, r in zip(trinity_scores, royal_scores)],
                 interpolate=True, color='red', alpha=0.2, label='Trinity Leading')

plt.fill_between(colombo_years, trinity_scores, royal_scores,
                 where=[r > t for t, r in zip(trinity_scores, royal_scores)],
                 interpolate=True, color='blue', alpha=0.2, label='Royal Leading')

plt.title('Bradby Scores in Colombo Leg Over Years', fontsize=16)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Leg Points', fontsize=12)
plt.xticks(rotation=45)
plt.grid(True, linestyle='--', alpha=0.3)
plt.legend(title='Team')
plt.tight_layout()
plt.show()
```



```
import matplotlib.pyplot as plt

# Prepare data for plotting
colombo_years = []
trinity_scores = []
royal_scores = []

for _, row in df.iterrows():
    if row['1st leg Venue'].lower() == 'colombo': # Filter for 1st leg in Colombo
        colombo_years.append(row['Year'])
        trinity_scores.append(row['Trinity 1st leg'])
        royal_scores.append(row['Royal 1st leg'])

plt.figure(figsize=(15,5))

# Plot lines
plt.plot(colombo_years, trinity_scores, color='red', marker='o', label='Trinity')
plt.plot(colombo_years, royal_scores, color='blue', marker='o', label='Royal')

# Shade area to show leading team
plt.fill_between(colombo_years, trinity_scores, royal_scores,
                 where=[t >= r for t, r in zip(trinity_scores, royal_scores)],
                 interpolate=True, color='red', alpha=0.2, label='Trinity Leading')

plt.fill_between(colombo_years, trinity_scores, royal_scores,
                 where=[r > t for t, r in zip(trinity_scores, royal_scores)],
                 interpolate=True, color='blue', alpha=0.2, label='Royal Leading')

plt.title('Bradby Scores in Colombo (1st Leg) Over Years', fontsize=16)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Leg Points', fontsize=12)
plt.xticks(rotation=45)
plt.grid(True, linestyle='--', alpha=0.3)
plt.legend(title='Team')
plt.tight_layout()
plt.show()
```