In [ ]:

```
from bs4 import BeautifulSoup
import requests
from csv import writer
from selenium import webdriver
from selenium.webdriver.support.ui import Select
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
c:\Users\mzuba\anaconda3\lib\site-packages\requests\__init__.py:89: RequestsDependencyWa
rning: urllib3 (1.26.9) or chardet (3.0.4) doesn't match a supported version!
  warnings.warn("urllib3 ({}) or chardet ({}) doesn't match a supported "
```

Data scraping/collecting

https://www.nba.com

In [ ]:

```
driver = webdriver.Firefox()
```

Analyzing traditional box score of two teams in the playoffs

In [ ]:

```
url = r"https://www.nba.com/stats/teams/boxscores-traditional/?Season=2021-22&SeasonTyp
driver.get(url)
```

In [ ]:

```
select = Select(driver.find_element_by_xpath(r"/html/body/main/div/div/div[2]/div/div/n
```

```
<ipython-input-5-b4dd7979dea8>:1: DeprecationWarning: find_element_by_xpath is deprecate
d. Please use find_element(by=By.XPATH, value=xpath) instead
  select = Select(driver.find_element_by_xpath(r"/html/body/main/div/div/div[2]/div/div/
nba-stat-table/div[1]/div/div/select"))
```

In [ ]:

```
select.select_by_index(0)
```

In [ ]:

```
src = driver.page_source
parser = BeautifulSoup(src, "lxml")
table = parser.find("div", attrs = {"class": "nba-stat-table__overflow"})
headers = table.findAll('th')
headerlist = [h.text.strip() for h in headers[1:]]
headerlist
```

Out[ ]:

```
['Match\xa0Up',
 'Game\xa0Date',
 'Season',
 'W/L',
 'MIN',
 'PTS',
 'FGM',
 'FGA',
 'FG%',
 '3PM',
 '3PA',
 '3P%',
 'FTM',
 'FTA',
 'FT%',
 'OREB',
 'DREB',
 'REB',
 'AST',
 'TOV',
 'STL',
 'BLK',
 'PF',
 '+/-']
```

166 Playoff games have been played so far in 2022

In [ ]:
```
rows = table.find_all('tr')[1:]
len(rows)
```

Out[ ]:
```
166
```

In [ ]:
```
player_stats = [[td.getText().strip() for td in rows[i].find_all('td')[1:]] for i in ra
player_stats[0]
```

Out[ ]:
```
['GSW vs. BOS',
 '06/05/2022',
 '',
 'W',
 '48',
 '107',
 '39',
 '86',
 '45.3',
 '15',
 '37',
 '40.5',
 '14',
 '20',
 '70.0',
 '6',
 '36',
 '42',
 '25',
 '12',
 '15',
 '2',
 '17',
 '19']
```

In [ ]:
```
df_stats = pd.DataFrame(player_stats, columns=headerlist)
df_stats.head()
```

Out[ ]:

	Match Up	Game Date	Season	W/L	MIN	PTS	FGM	FGA	FG%	3PM	...	FT%	OREB	DREB	RE
0	GSW vs. BOS	06/05/2022		W	48	107	39	86	45.3	15	...	70.0	6	36	4
1	BOS @ GSW	06/05/2022		L	48	88	30	80	37.5	15	...	76.5	6	37	4
2	BOS @ GSW	06/02/2022		W	48	120	43	85	50.6	21	...	81.3	7	32	3
3	GSW vs. BOS	06/02/2022		L	48	108	39	88	44.3	19	...	73.3	12	27	3
4	BOS @ MIA	05/29/2022		W	48	100	35	85	41.2	11	...	79.2	10	41	5

5 rows × 24 columns

In [ ]:
```
df_stats = df_stats.drop(['Season'], axis=1)
df_stats.head()
```

Out[ ]:

	Match Up	Game Date	W/L	MIN	PTS	FGM	FGA	FG%	3PM	3PA	...	FT%	OREB	DREB	REB
0	GSW vs. BOS	06/05/2022	W	48	107	39	86	45.3	15	37	...	70.0	6	36	42
1	BOS @ GSW	06/05/2022	L	48	88	30	80	37.5	15	37	...	76.5	6	37	43
2	BOS @ GSW	06/02/2022	W	48	120	43	85	50.6	21	41	...	81.3	7	32	39
3	GSW vs. BOS	06/02/2022	L	48	108	39	88	44.3	19	45	...	73.3	12	27	39
4	BOS @ MIA	05/29/2022	W	48	100	35	85	41.2	11	32	...	79.2	10	41	51

5 rows × 23 columns

Data visualization/analysis

In [ ]:
```
df_stats
```

Out[ ]:

	Match Up	Game Date	W/L	MIN	PTS	FGM	FGA	FG%	3PM	3PA	...	FT%	OREB	DREB	RE
0	GSW vs. BOS	06/05/2022	W	48	107	39	86	45.3	15	37	...	70.0	6	36	4
1	BOS @ GSW	06/05/2022	L	48	88	30	80	37.5	15	37	...	76.5	6	37	4
2	BOS @ GSW	06/02/2022	W	48	120	43	85	50.6	21	41	...	81.3	7	32	3
3	GSW vs. BOS	06/02/2022	L	48	108	39	88	44.3	19	45	...	73.3	12	27	3
4	BOS @ MIA	05/29/2022	W	48	100	35	85	41.2	11	32	...	79.2	10	41	5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
161	DEN @ GSW	04/16/2022	L	48	107	43	93	46.2	11	35	...	76.9	9	26	3
162	MEM vs. MIN	04/16/2022	L	48	117	39	86	45.3	7	27	...	74.4	8	27	3
163	PHI vs. TOR	04/16/2022	W	48	131	43	84	51.2	16	32	...	85.3	10	29	3
164	GSW vs. DEN	04/16/2022	W	48	123	43	82	52.4	16	35	...	72.4	10	31	4

	Match Up	Game Date	W/L	MIN	PTS	FGM	FGA	FG%	3PM	3PA	...	FT%	OREB	DREB	RE
165	DAL vs. UTA	04/16/2022	L	48	93	29	76	38.2	9	32	...	76.5	7	27	3

166 rows × 23 columns

Subsetting the dataset to only view playoff games of Celtics vs Warriors for their upcomming matchup in NBA finals

In [ ]:

```
nba_final_df = df_stats
nba_final_df
```

Out[ ]:

	Match Up	Game Date	W/L	MIN	PTS	FGM	FGA	FG%	3PM	3PA	...	FT%	OREB	DREB	RE
0	GSW vs. BOS	06/05/2022	W	48	107	39	86	45.3	15	37	...	70.0	6	36	4
1	BOS @ GSW	06/05/2022	L	48	88	30	80	37.5	15	37	...	76.5	6	37	4
2	BOS @ GSW	06/02/2022	W	48	120	43	85	50.6	21	41	...	81.3	7	32	3
3	GSW vs. BOS	06/02/2022	L	48	108	39	88	44.3	19	45	...	73.3	12	27	3
4	BOS @ MIA	05/29/2022	W	48	100	35	85	41.2	11	32	...	79.2	10	41	5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
161	DEN @ GSW	04/16/2022	L	48	107	43	93	46.2	11	35	...	76.9	9	26	3
162	MEM vs. MIN	04/16/2022	L	48	117	39	86	45.3	7	27	...	74.4	8	27	3
163	PHI vs. TOR	04/16/2022	W	48	131	43	84	51.2	16	32	...	85.3	10	29	3
164	GSW vs. DEN	04/16/2022	W	48	123	43	82	52.4	16	35	...	72.4	10	31	4
165	DAL vs. UTA	04/16/2022	L	48	93	29	76	38.2	9	32	...	76.5	7	27	3

166 rows × 23 columns

In [ ]:

```
nba_final_df = df_stats[df_stats['Match\xa0Up'].str.startswith('BO') | df_stats['Match\
df_stats['Match\xa0Up'].str.startswith('GS') | df_stats['Match\xa0Up'].str.endswith('SW
```

In [ ]:

```
nba_final_df
```

Out[ ]:

Match Up	Game Date	W/L	MIN	PTS	FGM	FGA	FG%	3PM	3PA	...	FT%	OREB	DREB	RE

	Match Up	Game Date	W/L	MIN	PTS	FGM	FGA	FG%	3PM	3PA	...	FT%	OREB	DREB	RE
0	GSW vs. BOS	06/05/2022	W	48	107	39	86	45.3	15	37	...	70.0	6	36	4
1	BOS @ GSW	06/05/2022	L	48	88	30	80	37.5	15	37	...	76.5	6	37	4
2	BOS @ GSW	06/02/2022	W	48	120	43	85	50.6	21	41	...	81.3	7	32	3
3	GSW vs. BOS	06/02/2022	L	48	108	39	88	44.3	19	45	...	73.3	12	27	3
4	BOS @ MIA	05/29/2022	W	48	100	35	85	41.2	11	32	...	79.2	10	41	5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
148	GSW vs. DEN	04/18/2022	W	48	126	46	84	54.8	17	40	...	70.8	6	29	3
156	BOS vs. BKN	04/17/2022	W	48	115	42	89	47.2	12	33	...	79.2	14	29	4
157	BKN @ BOS	04/17/2022	L	48	114	42	78	53.8	11	24	...	79.2	5	24	2
161	DEN @ GSW	04/16/2022	L	48	107	43	93	46.2	11	35	...	76.9	9	26	3
164	GSW vs. DEN	04/16/2022	W	48	123	43	82	52.4	16	35	...	72.4	10	31	4

72 rows × 23 columns

In [ ]:

```
boston_df = nba_final_df[nba_final_df['Match\xa0Up'].str.startswith('BO')]
gsw_df = nba_final_df[nba_final_df['Match\xa0Up'].str.startswith('GS')]
```

In [ ]:

```
boston_df
```

Out[ ]:

	Match Up	Game Date	W/L	MIN	PTS	FGM	FGA	FG%	3PM	3PA	...	FT%	OREB	DREB	RE
1	BOS @ GSW	06/05/2022	L	48	88	30	80	37.5	15	37	...	76.5	6	37	4
2	BOS @ GSW	06/02/2022	W	48	120	43	85	50.6	21	41	...	81.3	7	32	3
4	BOS @ MIA	05/29/2022	W	48	100	35	85	41.2	11	32	...	79.2	10	41	5
7	BOS vs. MIA	05/27/2022	L	48	103	32	72	44.4	11	33	...	90.3	6	28	3
10	BOS @ MIA	05/25/2022	W	48	93	33	71	46.5	10	33	...	73.9	6	40	4
15	BOS vs. MIA	05/23/2022	W	48	102	31	78	39.7	8	34	...	84.2	14	46	6

	Match Up	Game Date	W/L	MIN	PTS	FGM	FGA	FG%	3PM	3PA	...	FT%	OREB	DREB	RE
19	BOS vs. MIA	05/21/2022	L	48	103	34	70	48.6	12	32	...	76.7	9	35	4
23	BOS @ MIA	05/19/2022	W	48	127	43	84	51.2	20	40	...	91.3	8	33	4
26	BOS @ MIA	05/17/2022	L	48	107	36	79	45.6	11	34	...	75.0	8	30	3
28	BOS vs. MIL	05/15/2022	W	48	109	37	88	42.0	22	55	...	72.2	8	40	4
35	BOS @ MIL	05/13/2022	W	48	108	38	87	43.7	17	43	...	65.2	8	34	4
41	BOS vs. MIL	05/11/2022	L	48	107	42	82	51.2	10	31	...	86.7	5	31	3
51	BOS @ MIL	05/09/2022	W	48	116	42	84	50.0	14	37	...	90.0	4	34	3
59	BOS @ MIL	05/07/2022	L	48	101	32	87	36.8	9	33	...	82.4	11	36	4
70	BOS vs. MIL	05/03/2022	W	48	109	38	80	47.5	20	43	...	86.7	7	33	4
76	BOS vs. MIL	05/01/2022	L	48	89	28	84	33.3	18	50	...	75.0	10	38	4
101	BOS @ BKN	04/25/2022	W	48	116	42	89	47.2	14	35	...	81.8	10	35	4
113	BOS @ BKN	04/23/2022	W	48	109	42	84	50.0	12	39	...	92.9	13	21	3
134	BOS vs. BKN	04/20/2022	W	48	114	39	75	52.0	11	31	...	80.6	7	29	3
156	BOS vs. BKN	04/17/2022	W	48	115	42	89	47.2	12	33	...	79.2	14	29	4

20 rows × 23 columns

```
In [ ]: gsw_df
```

Out[ ]:

	Match Up	Game Date	W/L	MIN	PTS	FGM	FGA	FG%	3PM	3PA	...	FT%	OREB	DREB	RE
0	GSW vs. BOS	06/05/2022	W	48	107	39	86	45.3	15	37	...	70.0	6	36	4
3	GSW vs. BOS	06/02/2022	L	48	108	39	88	44.3	19	45	...	73.3	12	27	3
9	GSW vs. DAL	05/26/2022	W	48	120	45	88	51.1	14	36	...	100	14	37	5
13	GSW @ DAL	05/24/2022	L	48	109	41	84	48.8	10	28	...	65.4	6	36	4

	Match Up	Game Date	W/L	MIN	PTS	FGM	FGA	FG%	3PM	3PA	...	FT%	OREB	DREB	RE
17	GSW @ DAL	05/22/2022	W	48	109	38	81	46.9	11	32	...	88.0	14	33	4
20	GSW vs. DAL	05/20/2022	W	48	126	46	82	56.1	14	28	...	80.0	8	35	4
25	GSW vs. DAL	05/18/2022	W	48	112	46	82	56.1	10	29	...	62.5	7	44	5
34	GSW vs. MEM	05/13/2022	W	48	110	41	104	39.4	20	53	...	72.7	25	45	7
42	GSW @ MEM	05/11/2022	L	48	95	36	80	45.0	14	39	...	69.2	4	33	3
49	GSW vs. MEM	05/09/2022	W	48	101	36	90	40.0	9	37	...	90.9	10	44	5
57	GSW vs. MEM	05/07/2022	W	48	142	53	85	62.4	17	32	...	90.5	4	34	3
71	GSW @ MEM	05/03/2022	L	48	101	40	95	42.1	7	38	...	77.8	14	38	5
78	GSW @ MEM	05/01/2022	W	48	117	45	93	48.4	14	38	...	65.0	16	36	5
89	GSW vs. DEN	04/27/2022	W	48	102	37	82	45.1	13	31	...	71.4	6	31	3
110	GSW @ DEN	04/24/2022	L	48	121	43	86	50.0	12	34	...	71.9	9	27	3
129	GSW @ DEN	04/21/2022	W	48	118	41	74	55.4	18	40	...	64.3	4	26	3
148	GSW vs. DEN	04/18/2022	W	48	126	46	84	54.8	17	40	...	70.8	6	29	3
164	GSW vs. DEN	04/16/2022	W	48	123	43	82	52.4	16	35	...	72.4	10	31	4

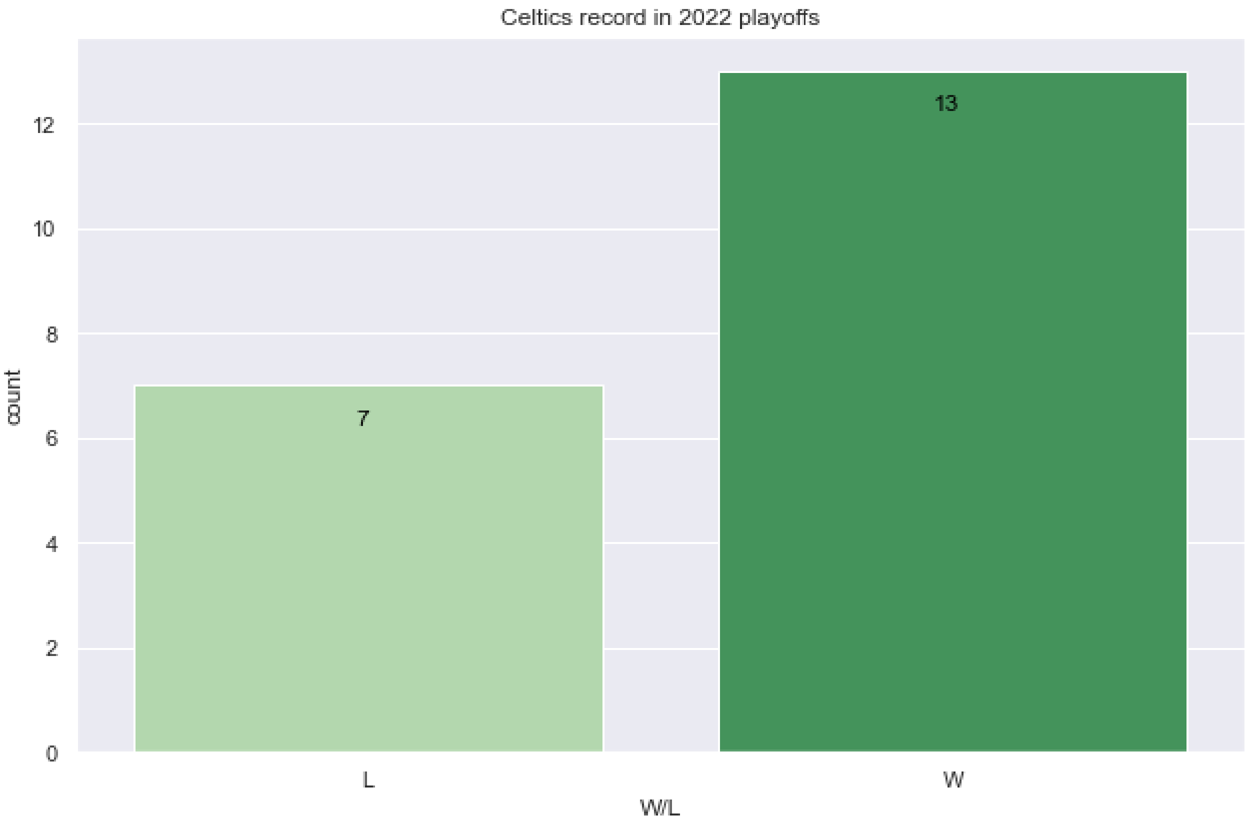
18 rows × 23 columns

What do each of these teams have to do to win the Nba finals?

Boston has won 13 out of their 20 games in playoffs. Their win ratio is 65%

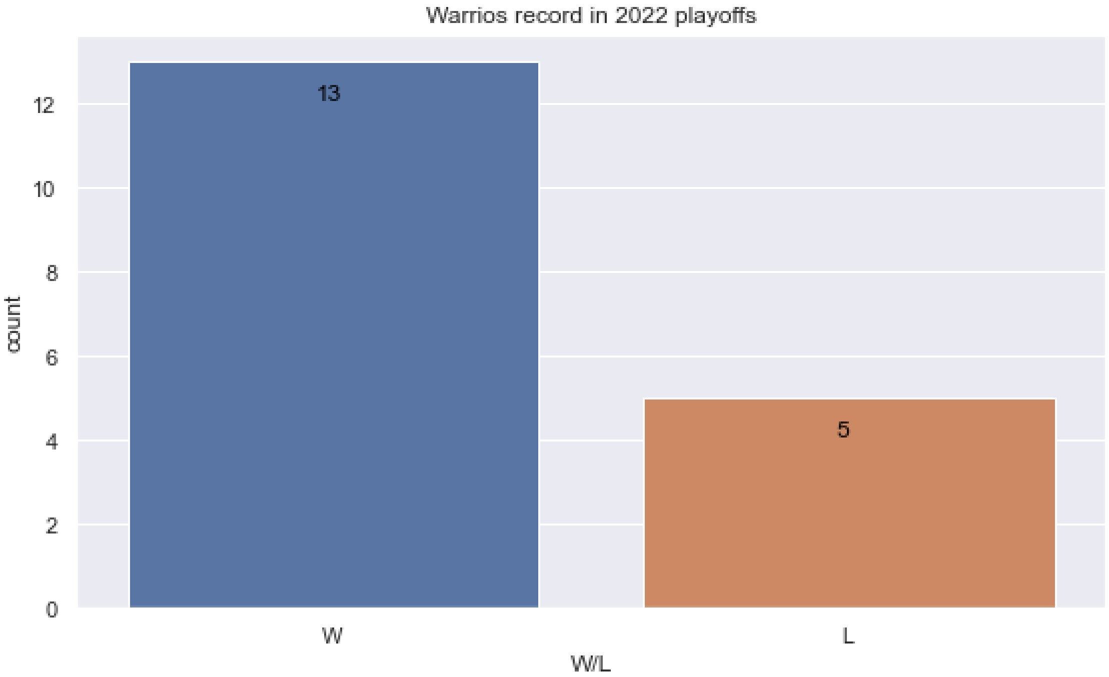
In [ ]:

```
# Lets first look at the amount of games each team won in playoffs
ax = sns.countplot(x='W/L', data=boston_df, palette="Greens")
for p in ax.patches:
    ax.annotate(f'\n{p.get_height()}', (p.get_x()+0.39, p.get_height()), ha='center', v
ax.set_title('Celtics record in 2022 playoffs')
sns.set(rc={'figure.figsize':(8,5)})
```



Warriors have won 13 out of their 18 games in playoffs. Their win ratio is 72%

```
In [ ]: # Lets first look at the amount of games each team won in playoffs
        ax = sns.countplot(x='W/L', data=gsw_df)
        for p in ax.patches:
            ax.annotate(f'\n{p.get_height()}', (p.get_x()+0.39, p.get_height()), ha='center', v
        ax.set_title('Warrios record in 2022 playoffs')
        sns.set(rc={'figure.figsize':(8,5)})
```



```
In [ ]: headerlist
```

Out[ ]:

```
['Match\xa0Up',
 'Game\xa0Date',
 'Season',
 'W/L',
 'MIN',
 'PTS',
 'FGM',
 'FGA',
 'FG%',
 '3PM',
 '3PA',
 '3P%',
 'FTM',
 'FTA',
 'FT%',
 'OREB',
 'DREB',
 'REB',
 'AST',
 'TOV',
 'STL',
 'BLK',
 'PF',
 '+/-']
```

In [ ]:

```python
boston_df['3P%'] = pd.to_numeric(boston_df['3P%'])
boston_df['PTS'] = pd.to_numeric(boston_df['PTS'])
boston_df['3PM'] = pd.to_numeric(boston_df['3PM'])
boston_df['FGM'] = pd.to_numeric(boston_df['FGM'])
boston_df['FG%'] = pd.to_numeric(boston_df['FG%'])
boston_df['FT%'] = pd.to_numeric(boston_df['FT%'])
boston_df['FTM'] = pd.to_numeric(boston_df['FTM'])
boston_df['TOV'] = pd.to_numeric(boston_df['TOV'])
boston_df['OREB'] = pd.to_numeric(boston_df['OREB'])
boston_df['DREB'] = pd.to_numeric(boston_df['DREB'])
boston_df['REB'] = pd.to_numeric(boston_df['REB'])
boston_df['STL'] = pd.to_numeric(boston_df['STL'])
boston_df['BLK'] = pd.to_numeric(boston_df['BLK'])
boston_df['PF'] = pd.to_numeric(boston_df['PF'])
boston_df.dtypes
```

```
<ipython-input-22-595adae23d13>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  boston_df['3P%'] = pd.to_numeric(boston_df['3P%'])
<ipython-input-22-595adae23d13>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  boston_df['PTS'] = pd.to_numeric(boston_df['PTS'])
<ipython-input-22-595adae23d13>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  boston_df['3PM'] = pd.to_numeric(boston_df['3PM'])
<ipython-input-22-595adae23d13>:4: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  boston_df['FGM'] = pd.to_numeric(boston_df['FGM'])
<ipython-input-22-595adae23d13>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  boston_df['FG%'] = pd.to_numeric(boston_df['FG%'])
<ipython-input-22-595adae23d13>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  boston_df['FT%'] = pd.to_numeric(boston_df['FT%'])
<ipython-input-22-595adae23d13>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  boston_df['FTM'] = pd.to_numeric(boston_df['FTM'])
<ipython-input-22-595adae23d13>:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  boston_df['TOV'] = pd.to_numeric(boston_df['TOV'])
<ipython-input-22-595adae23d13>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  boston_df['OREB'] = pd.to_numeric(boston_df['OREB'])
<ipython-input-22-595adae23d13>:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  boston_df['DREB'] = pd.to_numeric(boston_df['DREB'])
<ipython-input-22-595adae23d13>:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  boston_df['REB'] = pd.to_numeric(boston_df['REB'])
<ipython-input-22-595adae23d13>:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  boston_df['STL'] = pd.to_numeric(boston_df['STL'])
<ipython-input-22-595adae23d13>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  boston_df['BLK'] = pd.to_numeric(boston_df['BLK'])
<ipython-input-22-595adae23d13>:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  boston_df['PF'] = pd.to_numeric(boston_df['PF'])
```

Out[ ]:

```
Match Up      object
Game Date     object
W/L           object
MIN           object
PTS            int64
FGM            int64
FGA           object
FG%          float64
3PM            int64
3PA           object
3P%          float64
FTM            int64
FTA           object
FT%          float64
OREB           int64
DREB           int64
REB            int64
AST           object
TOV            int64
STL            int64
BLK            int64
PF             int64
+/-           object
dtype: object
```

In [ ]:

```
gsw_df['3P%'] = pd.to_numeric(gsw_df['3P%'])
gsw_df['3PM'] = pd.to_numeric(gsw_df['3PM'])
gsw_df['PTS'] = pd.to_numeric(gsw_df['PTS'])
gsw_df['FG%'] = pd.to_numeric(gsw_df['FG%'])
gsw_df['FGM'] = pd.to_numeric(gsw_df['FGM'])
gsw_df['FTM'] = pd.to_numeric(gsw_df['FTM'])
gsw_df['FT%'] = pd.to_numeric(gsw_df['FT%'])
gsw_df['TOV'] = pd.to_numeric(gsw_df['TOV'])
gsw_df['OREB'] = pd.to_numeric(gsw_df['OREB'])
gsw_df['DREB'] = pd.to_numeric(gsw_df['DREB'])
gsw_df['REB'] = pd.to_numeric(gsw_df['REB'])
gsw_df['STL'] = pd.to_numeric(gsw_df['STL'])
gsw_df['BLK'] = pd.to_numeric(gsw_df['BLK'])
gsw_df['PF'] = pd.to_numeric(gsw_df['PF'])
gsw_df.dtypes
```

```
<ipython-input-78-83be22798500>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  gsw_df['3P%'] = pd.to_numeric(gsw_df['3P%'])
<ipython-input-78-83be22798500>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  gsw_df['3PM'] = pd.to_numeric(gsw_df['3PM'])
<ipython-input-78-83be22798500>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  gsw_df['PTS'] = pd.to_numeric(gsw_df['PTS'])
<ipython-input-78-83be22798500>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  gsw_df['FG%'] = pd.to_numeric(gsw_df['FG%'])
<ipython-input-78-83be22798500>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  gsw_df['FGM'] = pd.to_numeric(gsw_df['FGM'])
<ipython-input-78-83be22798500>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  gsw_df['FTM'] = pd.to_numeric(gsw_df['FTM'])
<ipython-input-78-83be22798500>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  gsw_df['FT%'] = pd.to_numeric(gsw_df['FT%'])
<ipython-input-78-83be22798500>:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  gsw_df['TOV'] = pd.to_numeric(gsw_df['TOV'])
<ipython-input-78-83be22798500>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  gsw_df['OREB'] = pd.to_numeric(gsw_df['OREB'])
<ipython-input-78-83be22798500>:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  gsw_df['DREB'] = pd.to_numeric(gsw_df['DREB'])
<ipython-input-78-83be22798500>:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
```

```
  gsw_df['REB'] = pd.to_numeric(gsw_df['REB'])
<ipython-input-78-83be22798500>:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  gsw_df['STL'] = pd.to_numeric(gsw_df['STL'])
<ipython-input-78-83be22798500>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  gsw_df['BLK'] = pd.to_numeric(gsw_df['BLK'])
<ipython-input-78-83be22798500>:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  gsw_df['PF'] = pd.to_numeric(gsw_df['PF'])
```

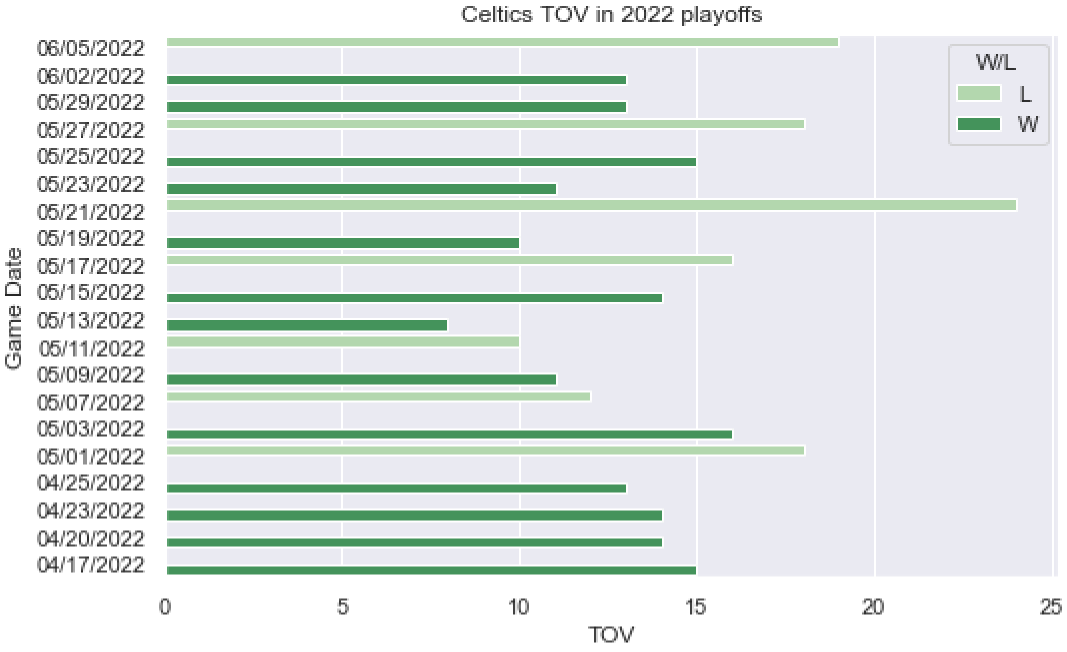
Out[ ]:

```
Match Up      object
Game Date     object
W/L           object
MIN           object
PTS            int64
FGM            int64
FGA           object
FG%          float64
3PM            int64
3PA           object
3P%          float64
FTM            int64
FTA           object
FT%          float64
OREB           int64
DREB           int64
REB            int64
AST           object
TOV            int64
STL            int64
BLK            int64
PF             int64
+/-           object
dtype: object
```

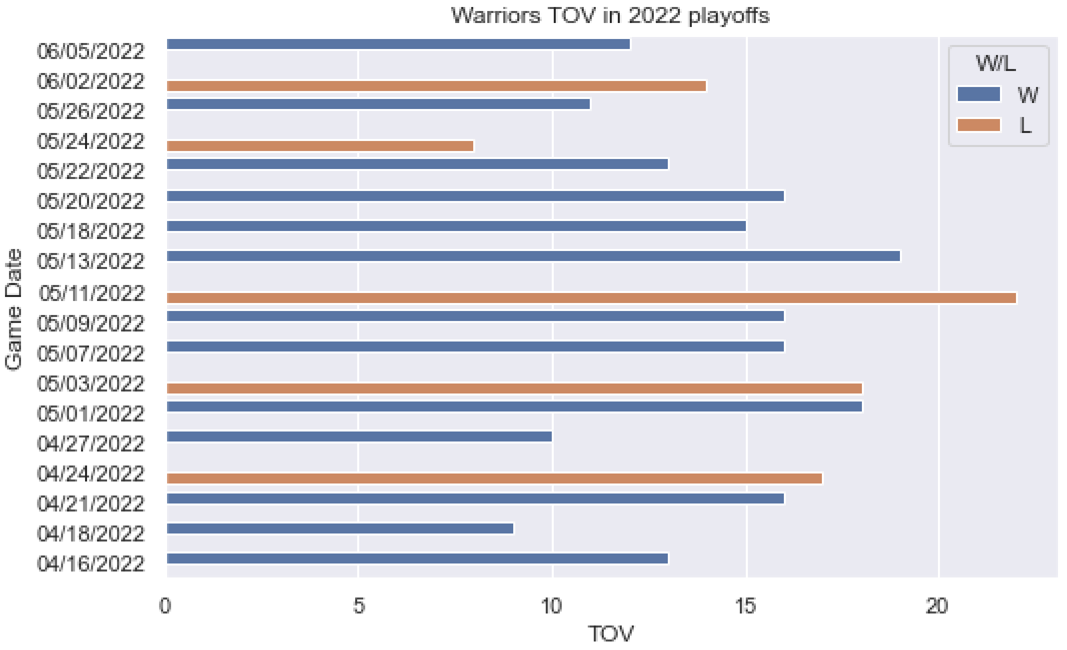
How can Celtics or Warrios win the NBA finals?

In [ ]:

```
ax = sns.barplot(x= 'TOV', y="Game\xa0Date", hue="W/L", data=boston_df, palette = "Gree
sns.set(rc={'figure.figsize':(8,5)})
```

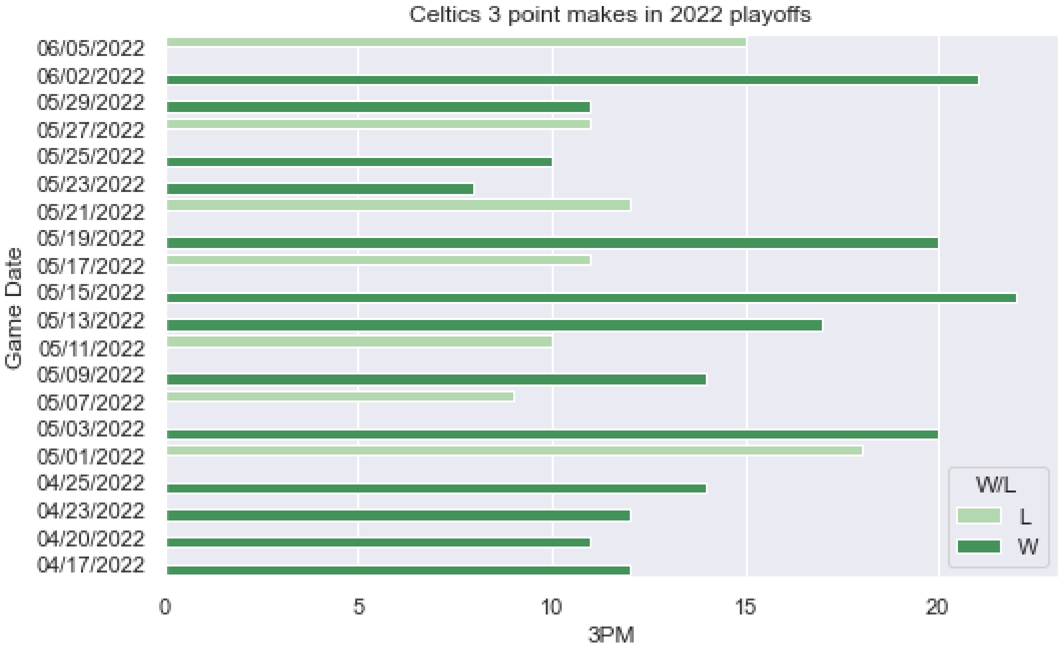


```
In [ ]: ax = sns.barplot(x= 'TOV', y="Game\xa0Date", hue="W/L", data=gsw_df).set(title='Warrior
        sns.set(rc={'figure.figsize':(8,5)})
```

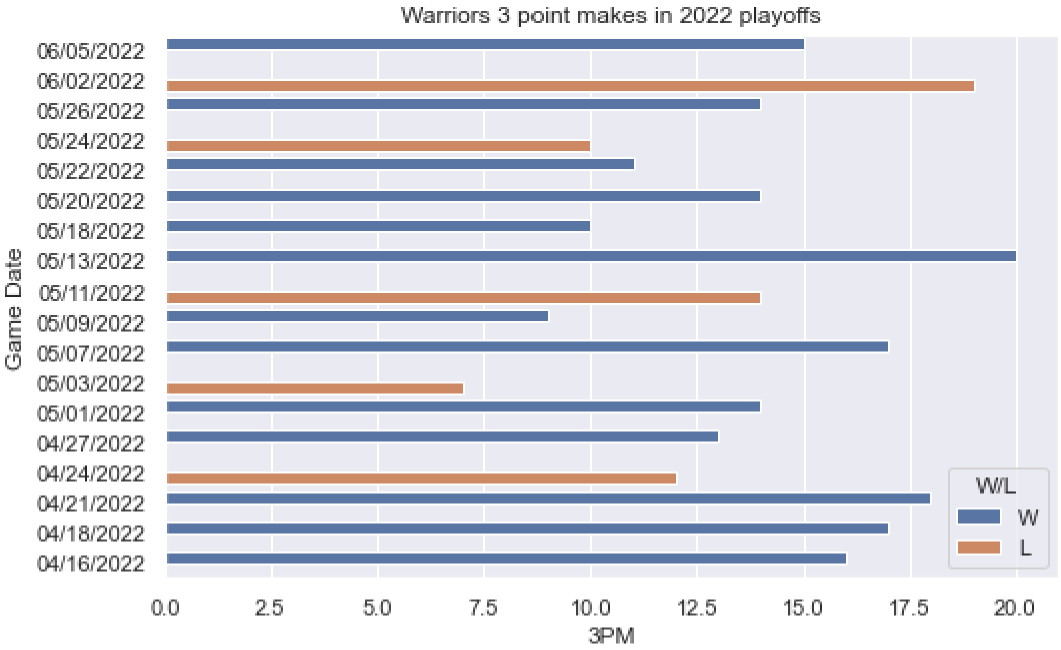


As seen from the plots above, both teams have lost most of their games when being careless with the ball. It is important for the defense of both teams to throw active hands on the ball to try and force timely turnovers.

```
In [ ]: ax = sns.barplot(x= '3PM', y="Game\xa0Date", hue="W/L", data=boston_df, palette = "Gree
        sns.set(rc={'figure.figsize':(8,5)})
```

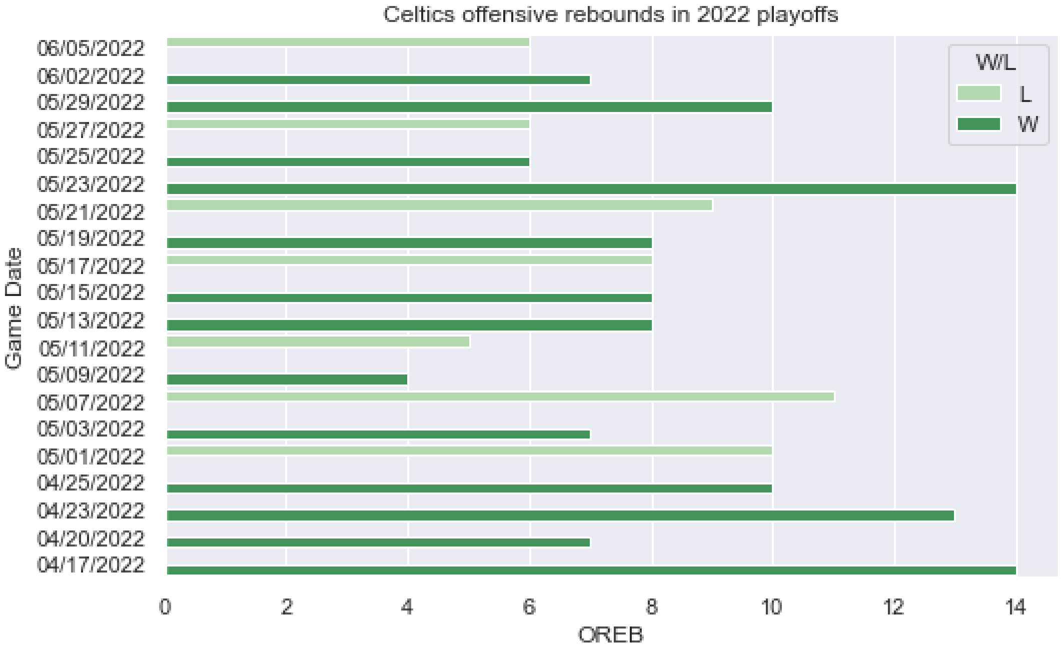


In [ ]:
```
ax = sns.barplot(x= '3PM', y="Game\xa0Date", hue="W/L", data=gsw_df).set(title='Warrior
sns.set(rc={'figure.figsize':(8,5)})
```



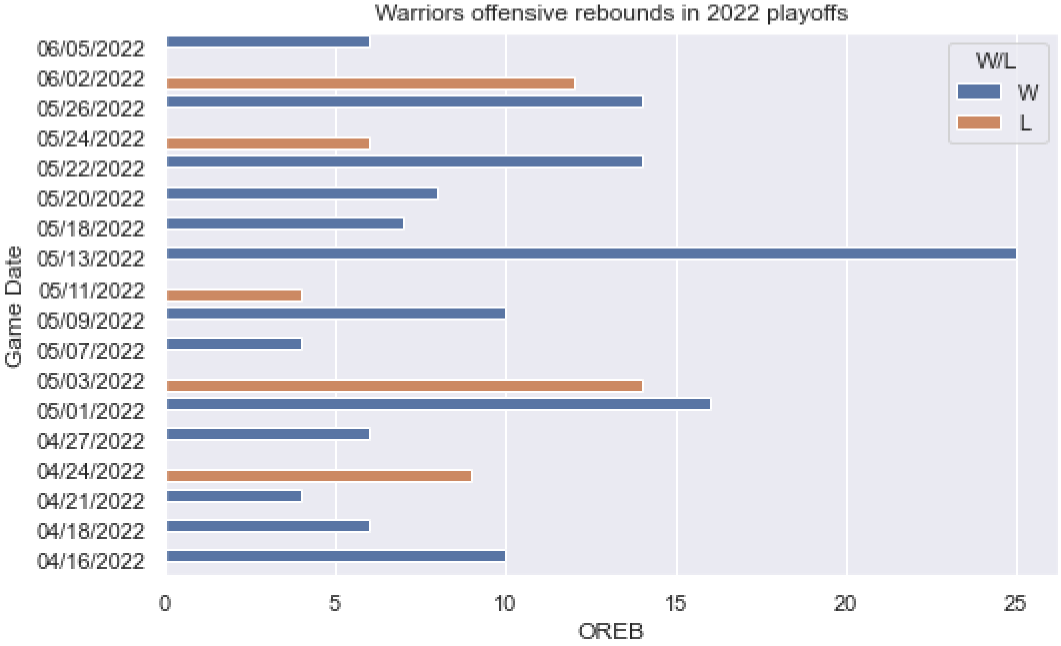
3 point makes impacted the outcome of game throughout the playoffs for both teams.....except when they faced each other and broke the pattern by making almost the same amount of threes.

In [ ]:
```
#plt.rcParams["figure.figsize"] = [12, 7]
#plt.rcParams["figure.autolayout"] = True
#f, axes = plt.subplots(1, 2)
ax = sns.barplot(x= 'OREB', y="Game\xa0Date", hue="W/L", data=boston_df, palette = "Gre
sns.set(rc={'figure.figsize':(8,5)})
```



In [ ]:

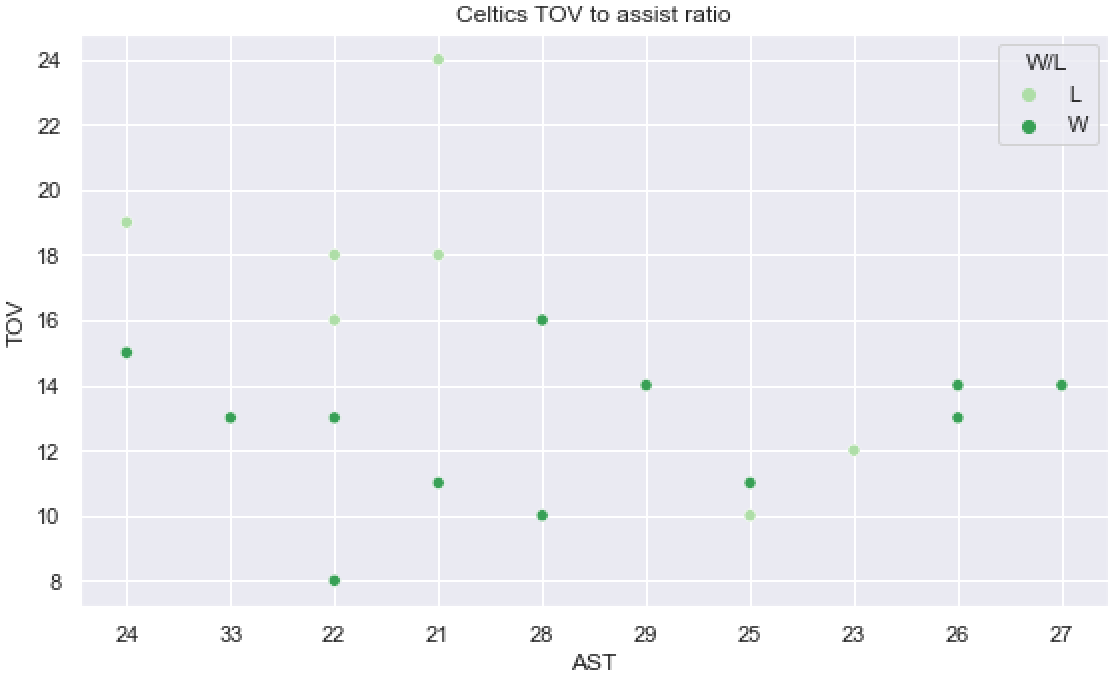
```
#plt.rcParams["figure.figsize"] = [12, 7]
#plt.rcParams["figure.autolayout"] = True
#f, axes = plt.subplots(1, 2)
sns.barplot(x= 'OREB', y="Game\xa0Date", hue="W/L", data=gsw_df).set(title='Warriors of
sns.set(rc={'figure.figsize':(8,5)})
```



Another strategy these two teams can use is playing the lineups that can acquire them offensive rebounds = second chance pts oppertunity = Wins.

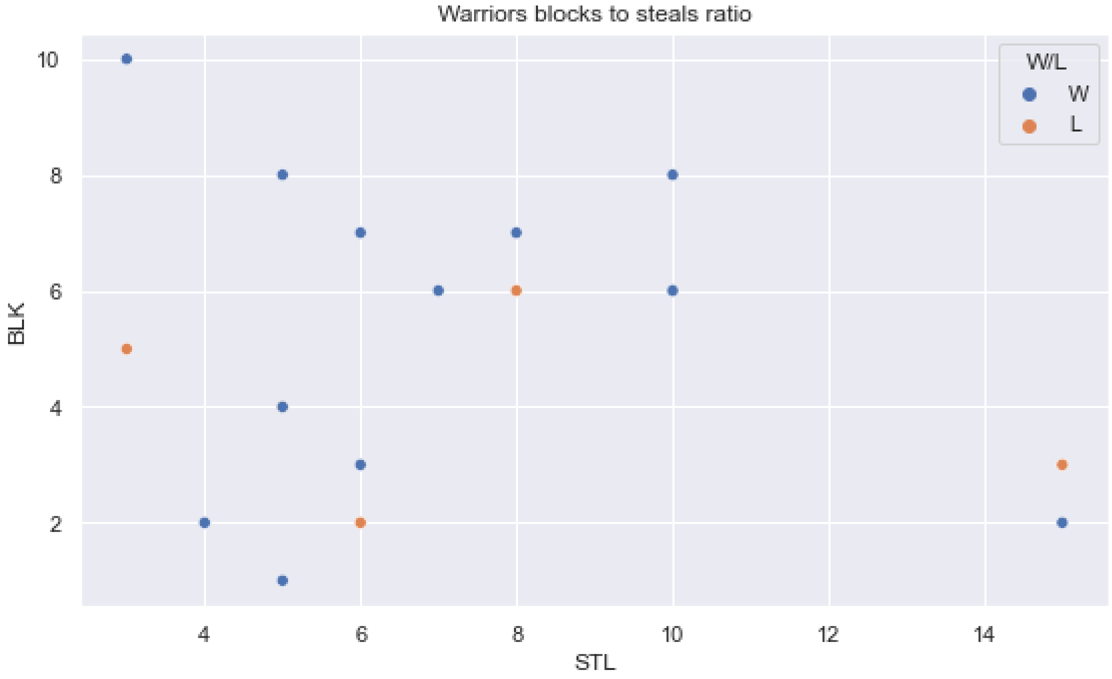
In [ ]:

```
ax = sns.scatterplot(data=boston_df, x="AST", y="TOV", hue='W/L', palette="Greens").set
sns.set(rc={'figure.figsize':(8,5)})
```



Celtics have lost all the games where their turnovers exceeded 16

```
In [ ]: ax = sns.scatterplot(data=gsw_df, x="STL", y="BLK", hue='W/L').set(title='Warriors bloc
        sns.set(rc={'figure.figsize':(8,5)})
```



Defensive stops is key for warriors because they have won most of their games when exceeding 6 blocks

```
In [ ]: labels_corr = ['PTS',
         'FGM',
         'FG%',
         '3PM',
         '3P%',
         'FTM',
         'FT%',
         'OREB',
         'DREB',
    'AST',
    'TOV',
    'STL',
    'BLK',
    'PF',
    'W/L']

df_corr_boston = pd.DataFrame(boston_df,columns=labels_corr)
df_corr_gsw = pd.DataFrame(gsw_df,columns=labels_corr)
df_corr_gsw.head()
```

Out[ ]:

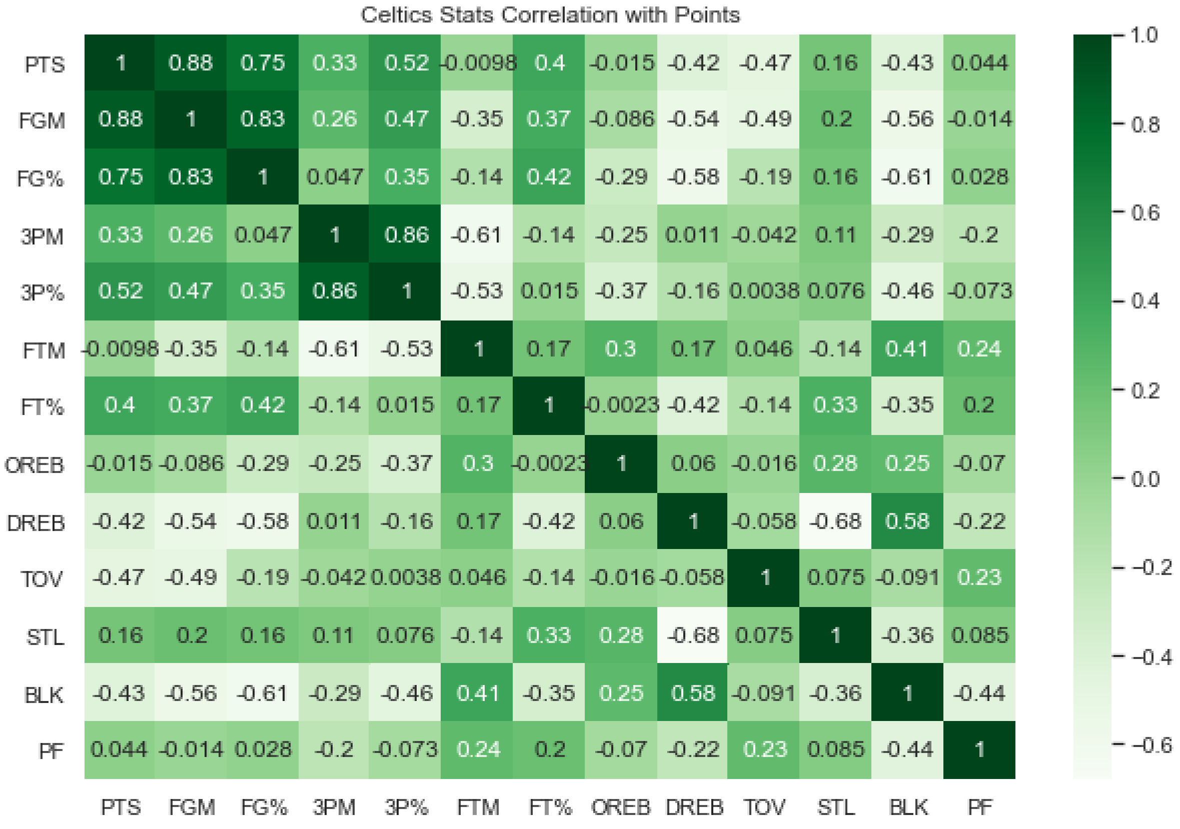
	PTS	FGM	FG%	3PM	3P%	FTM	FT%	OREB	DREB	AST	TOV	STL	BLK	PF	W/L
0	107	39	45.3	15	40.5	14	70.0	6	36	25	12	15	2	17	W
3	108	39	44.3	19	42.2	11	73.3	12	27	24	14	8	6	16	L
9	120	45	51.1	14	38.9	16	100.0	14	37	36	11	6	3	20	W
13	109	41	48.8	10	35.7	17	65.4	6	36	26	8	5	4	18	L
17	109	38	46.9	11	34.4	22	88.0	14	33	28	13	5	1	23	W

In [ ]:

```
plt.rcParams["figure.figsize"] = [9, 6]
plt.rcParams["figure.autolayout"] = True
corrMatrix = df_corr_boston.corr()
ax = sns.heatmap(corrMatrix, annot=True, cmap="Greens")
ax.set_title('Celtics Stats Correlation with Points')
```

Out[ ]:

```
Text(0.5, 1.0, 'Celtics Stats Correlation with Points')
```

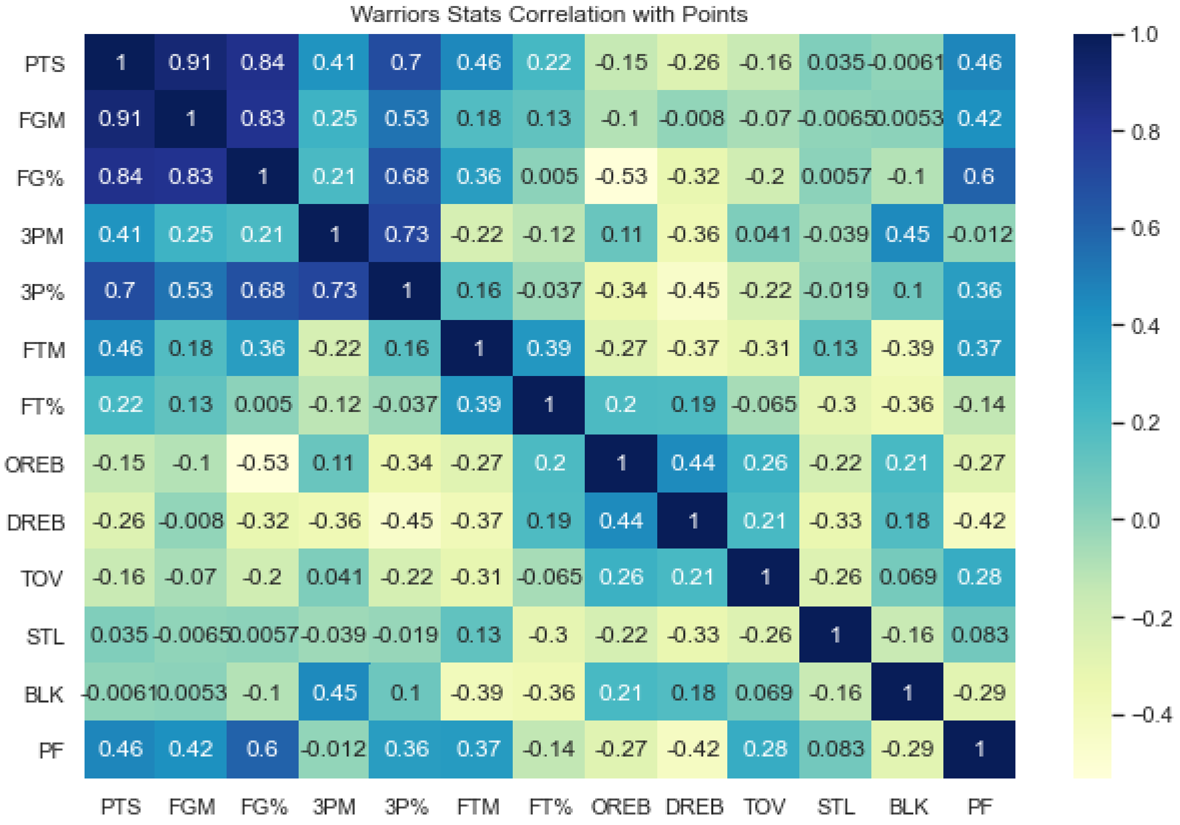


In [ ]:

```
plt.rcParams["figure.figsize"] = [9, 6]
plt.rcParams["figure.autolayout"] = True
```

```python
corrMatrix = df_corr_gsw.corr()
ax = sns.heatmap(corrMatrix, annot=True, cmap="YlGnBu")
ax.set_title('Warriors Stats Correlation with Points')
```

```
Out[ ]: Text(0.5, 1.0, 'Warriors Stats Correlation with Points')
```



	PTS	FGM	FG%	3PM	3P%	FTM	FT%	OREB	DREB	TOV	STL	BLK	PF
PTS	1	0.91	0.84	0.41	0.7	0.46	0.22	-0.15	-0.26	-0.16	0.035	-0.0061	0.46
FGM	0.91	1	0.83	0.25	0.53	0.18	0.13	-0.1	-0.008	-0.07	-0.0065	0.0053	0.42
FG%	0.84	0.83	1	0.21	0.68	0.36	0.005	-0.53	-0.32	-0.2	0.0057	-0.1	0.6
3PM	0.41	0.25	0.21	1	0.73	-0.22	-0.12	0.11	-0.36	0.041	-0.039	0.45	-0.012
3P%	0.7	0.53	0.68	0.73	1	0.16	-0.037	-0.34	-0.45	-0.22	-0.019	0.1	0.36
FTM	0.46	0.18	0.36	-0.22	0.16	1	0.39	-0.27	-0.37	-0.31	0.13	-0.39	0.37
FT%	0.22	0.13	0.005	-0.12	-0.037	0.39	1	0.2	0.19	-0.065	-0.3	-0.36	-0.14
OREB	-0.15	-0.1	-0.53	0.11	-0.34	-0.27	0.2	1	0.44	0.26	-0.22	0.21	-0.27
DREB	-0.26	-0.008	-0.32	-0.36	-0.45	-0.37	0.19	0.44	1	0.21	-0.33	0.18	-0.42
TOV	-0.16	-0.07	-0.2	0.041	-0.22	-0.31	-0.065	0.26	0.21	1	-0.26	0.069	0.28
STL	0.035	-0.0065	0.0057	-0.039	-0.019	0.13	-0.3	-0.22	-0.33	-0.26	1	-0.16	0.083
BLK	-0.0061	0.0053	-0.1	0.45	0.1	-0.39	-0.36	0.21	0.18	0.069	-0.16	1	-0.29
PF	0.46	0.42	0.6	-0.012	0.36	0.37	-0.14	-0.27	-0.42	0.28	0.083	-0.29	1

It was self explanatory that FGM will have the most correlation with points. However, we do see that warrios 3PM has more impact on points than celtics 3PM. Furthemore, factors like turnovers can decrease the points of a team. Celtics have to be more careful with the ball because their turnovers has more impact on their points than the warriors turnovers. This does make sense because a hot striking team like the warriors can surely decrease the impact of a turnover from their 3PM.

Analysis summary:

- Both teams have to rely on their 3PM to have a chance at winning the game

- Second chance points from offensive rebounds can give both teams an oppertunity to win the game

- Celtics have to limit their turonvers < 15, to have a chance at winning the game

- Warriors have to exectute defensive stops from blocks, steals to have a chance at winning the game

```
In [ ]:
```