The Challenge

- How many matches were played each world cup year from 1930.
- Total goals scored for each tournament year.
- All teams who have reached finals and how many times.
- All teams who have reached semis and how many times.
- How many goals and average goals scored in all semi-finals.
- How many goals and average goals scored in all quarter-finals.
- How many goals and average number scored in all finals
- How many matches were played outside quarter-finals and above.
- **The #kicker**:
- Two new columns for each of the outcome of every match stating:
- a. outcome = D for Draw, A for AwayTeam Wins, H for HomeTeam wins.
- b. Winner of each game: 'Draw' if no winner.

PS: Our predominant choice of plotting library on this one is plotly

In [1]:
```
import pandas as pd
import plotly.express as px
import numpy as np
```

In [2]:
```
#Read our dataset
df = pd.read_csv("world_cup_results.csv")
```

In [3]:
```
#Initial of our dataframe
df.shape
```

Out[3]:
```
(852, 11)
```

In [4]:
```
#Some information about our df shows we have no missing values
#However our date field is not of the right type. We'll deal with it as we go
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 852 entries, 0 to 851
Data columns (total 11 columns):
Year           852 non-null int64
Date           852 non-null object
Time           852 non-null object
Round          852 non-null object
Stadium        852 non-null object
City           852 non-null object
HomeTeam       852 non-null object
HomeGoals      852 non-null int64
AwayGoals      852 non-null int64
AwayTeam       852 non-null object
Observation    852 non-null object
dtypes: int64(3), object(8)
memory usage: 73.3+ KB
```

In [5]:
```
#Dependinf on your datasource, you might have column names with spaces
#One quick thing I do most times is to convert everything to lowercase and add undr
#It's not the case here but I will drop in the flow all the same

# headers = [line.lower().replace(' ', '_') for line in df.columns]
# df.columns = headers
# df.head()
```

In [6]:
```
#let's deal with the duplicates
#Notice that from a shape of (852cols, 11rows) we not arrive at (836, 11). There wa
df = df.drop_duplicates()
df.shape
```

Out[6]: (836, 11)

Let's prep our data in antipaction of questions lined up

- We need a column for total goals scores combining home and away goals for each row/match
- We wanna isolate the day of the week and month from the date column making them columns of their own.

In [7]:
```
#Total goals columns
df['TotalGoals'] = df['HomeGoals'] + df['AwayGoals']
```

In [8]:
```
#get a day of the week and month columns
df['month'] = df['Date'].apply(lambda x: x.split('-')[1]) #split the literal string
df['day'] = pd.to_datetime(df['Date']).dt.day_name() #convert to datetime and get d
```

In [9]:
```
#Let's see what our df looks like now by peeping the head
df.head(2)
```

Out[9]:

	Year	Date	Time	Round	Stadium	City	HomeTeam	HomeGoals	AwayGoals	AwayTeam	Obser
0	1930	13-Jul-30	15:00	Group 1	Pocitos	Montevideo	France	4	1	Mexico	
1	1930	13-Jul-30	15:00	Group 4	Parque Central	Montevideo	USA	3	0	Belgium	

In [10]:
```
#For the kicker down the line we are going to add two more columns
#I like to see all my cols if possible so in anticipation of the 2 coming up, I'll
#Notice that in this process  I have taken the liberty to order the columns
df = df[['Year', 'month', 'day', 'Time', 'Round', 'HomeTeam', 'HomeGoals', 'AwayTea
df.head(2)
```

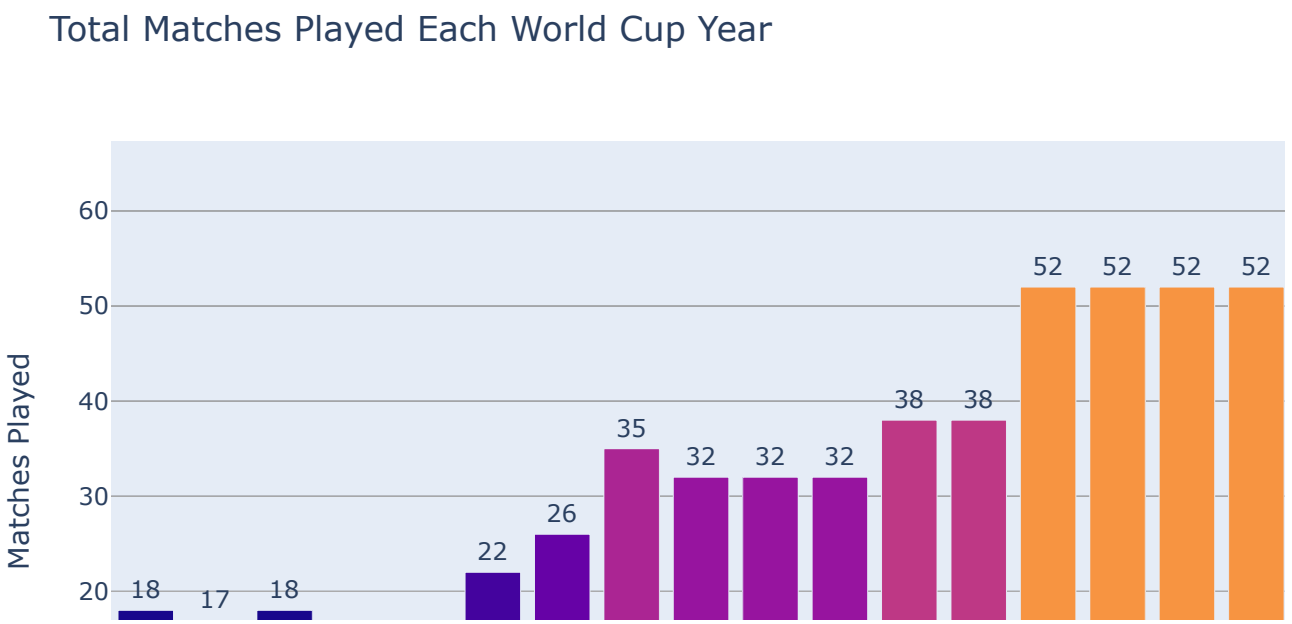
Out[10]:

	Year	month	day	Time	Round	HomeTeam	HomeGoals	AwayTeam	AwayGoals	TotalGoals
0	1930	Jul	Saturday	15:00	Group 1	France	4	Mexico	1	5
1	1930	Jul	Saturday	15:00	Group 4	USA	3	Belgium	0	3

Que 1: How many matches were played each world cup year from 1930.

In [11]:

```
#A value_count on the Year column nicely delivers this
#To plot this effortlessly with plotly, we will convert the result to a fresh dataf
#Nothing how nicely plotly highlights the expected years world cups were not played
matches_per_year = df.Year.value_counts() #a series to hold our values
all_games = pd.DataFrame(matches_per_year) #make series into a dataframe
all_games.reset_index(inplace=True) #reset it's index inplace
all_games.columns = ['Year', 'Matches'] #rename the columns as needed
fig = px.bar(all_games, x='Year', y='Matches', text='Matches', color='Matches', hei
             labels={'Matches':'Matches Played', 'Year':'World Cup Year'},
            title="Total Matches Played Each World Cup Year")
fig.update_traces(texttemplate='%{text}', textposition='outside')
fig.update_layout(uniformtext_minsize=8, uniformtext_mode='hide')
fig.update_xaxes(
    tickangle=45, tickfont=dict(family='Arial', color='blue', size=14),
    tickvals=[line for line in all_games.Year])
fig.show()
```



Salient Point 0

- My first salient observation is that there was no FIFA world cup in 1942 abd 1946 as a result of World War II

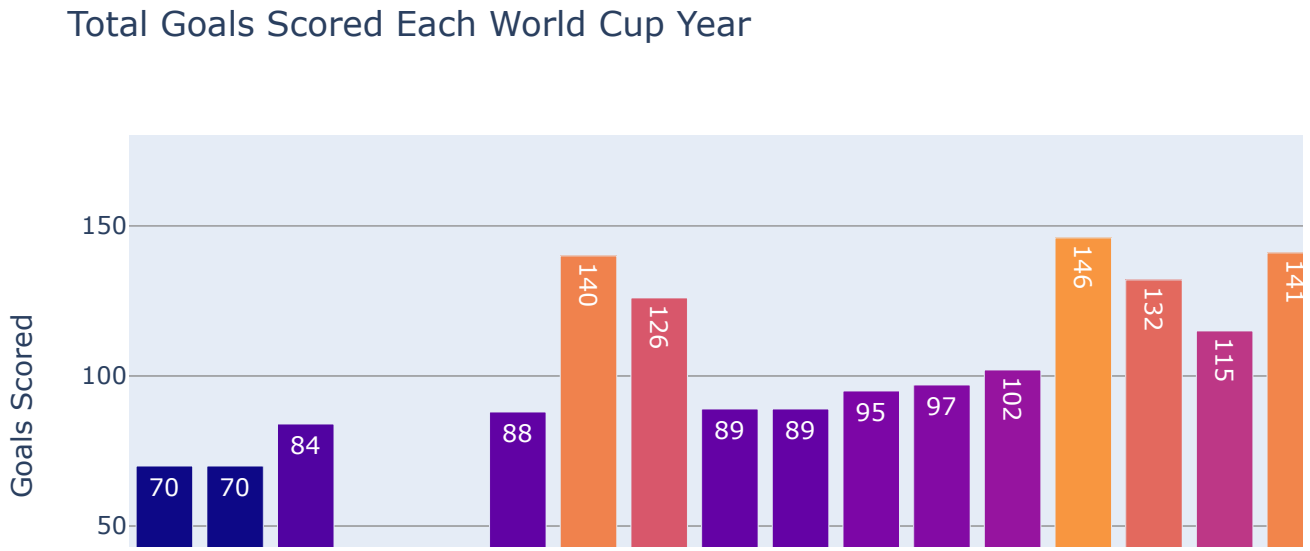
Que 2: Total goals scored for each tournament year.

- Group initial dataframe by year summing total goals
- Convert result to a dataframe and drop a plot on the result

In [12]:

```
all_goals = df.groupby(['Year']).TotalGoals.sum()
all_goals_df = pd.DataFrame(all_goals)
all_goals_df.reset_index(inplace=True) #reset it's index inplace
all_goals_df.columns = ['Year', 'Goals'] #rename the columns as needed

fig = px.bar(all_goals_df, x='Year', y='Goals', text='Goals', color='Goals', height
             labels={'Goals':'Goals Scored', 'Year':'World Cup Year'},
            title="Total Goals Scored Each World Cup Year")
fig.update_traces(texttemplate='%{text}', textposition='inside')
fig.update_layout(uniformtext_minsize=8, uniformtext_mode='hide')
fig.update_xaxes(
    tickangle=45, tickfont=dict(family='Arial', color='blue', size=14),
    tickvals=[line for line in all_goals_df.Year])
fig.show()
```



Salient Point 1

- My second salient observation is that though it had less macthes played (at 26) than subsequent 5 world cups after it, the net shook more times in 1954 than them these 5 individually. That's remarkable. They either had terrible goalkeepers and/or defenders or strikers of that year were prolific amongs other considerations.

Que 3: All teams who have reached finals and how many times.

In [13]:
```
#Take a piece of the df corresponding to all 'Final' in the 'Round' column
all_finals = df[df['Round'] == 'Final']
all_finals.head()
```

Out[13]:

	Year	month	day	Time	Round	HomeTeam	HomeGoals	AwayTeam	AwayGoals	TotalGoals
17	1930	Jul	Tuesday	14:15	Final	Uruguay	4	Argentina	2	6
34	1934	Jun	Saturday	17:30	Final	Italy	2	Czechoslovakia	1	3
52	1938	Jun	Saturday	17:00	Final	Italy	4	Hungary	2	6
100	1954	Jul	Saturday	17:00	Final	Germany FR	3	Hungary	2	5
135	1958	Jun	Saturday	15:00	Final	Brazil	5	Sweden	2	7

In [14]:
```
#Let's make a list of all teams who reach this stage.
#This will be a list of all featuring HomeTeams and AwayTeams
#A simple concatenation of a python list of both will do
teams = [line for line in all_finals.HomeTeam] + [line for line in all_finals.AwayT
#Peep a sample
teams[:5]
```

Out[14]:
```
['Uruguay', 'Italy', 'Italy', 'Germany FR', 'Brazil']
```

In [15]:
```
#To count the frequency that I am going to eventually plot I prefer to use a datafr
#So I will make a dataframe from the list 'teams' and take a drop a value_counts().
all_finals_df = pd.DataFrame(columns=['Teams'], data = teams)
#peep the head()
all_finals_df.head(2)
```

Out[15]:

	Teams
0	Uruguay
1	Italy

In [16]:
```
#To demonstrate this value_counts() counting, see the result before we plot
#PS: I am choosing to leave Germany and Germany FR as different entities
all_finals_df.Teams.value_counts()
```

Out[16]:
```
Brazil           6
Germany FR       6
Italy            6
Argentina        5
Netherlands      3
Hungary          2
Czechoslovakia   2
Germany          2
France           2
England          1
Sweden           1
Spain            1
Uruguay          1
Name: Teams, dtype: int64
```

In [17]:
```
#Ler me get this same result outside pandas
finals_teams = {} #Dict to team as key and apperance number as value
for team in teams:
    if team in finals_teams.keys(): #if this team is already in the dict
        finals_teams[team] += 1 #increment it by one
    else: #else
        finals_teams[team] = 1 #It's its first instance in the loop, assign it a va

finals_teams
```

Out[17]:
```
{'Uruguay': 1,
 'Italy': 6,
 'Germany FR': 6,
 'Brazil': 6,
 'England': 1,
 'Netherlands': 3,
 'Argentina': 5,
 'Germany': 2,
 'Czechoslovakia': 2,
 'Hungary': 2,
 'Sweden': 1,
 'France': 2,
 'Spain': 1}
```

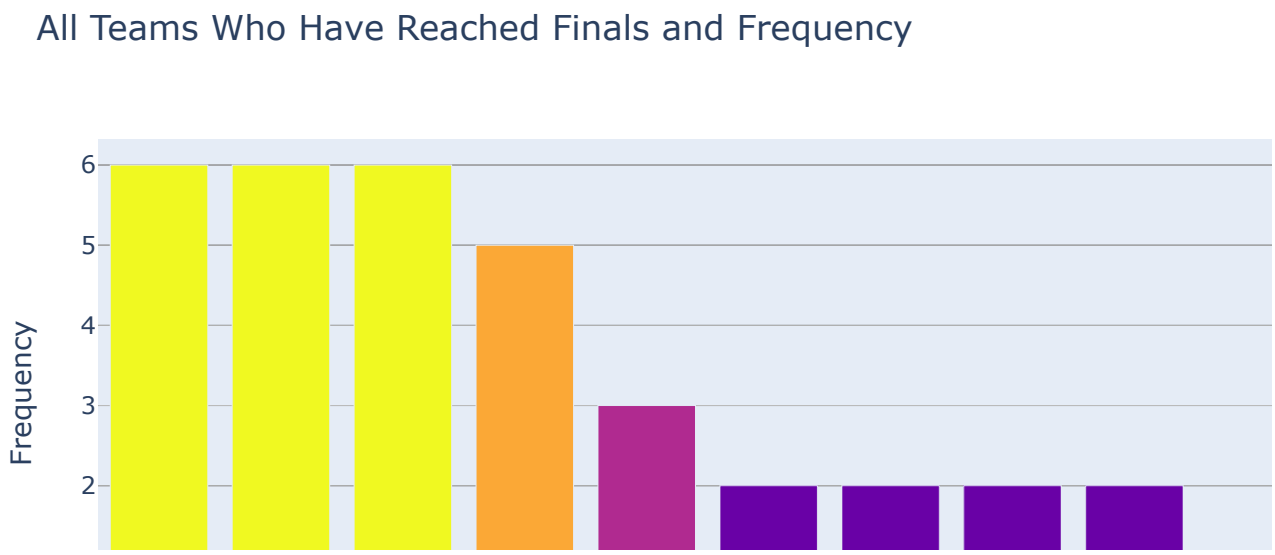
In [18]:
```
#We know that a python dictionary as a property is unordered so we can't successful
#The above is not sorted so let's sort and reverse to get it in descending order.
#Notice that the result is a list of tuples. To plot take not of this.
#Compare the result below with what value_counts() gave us some cells up.
#I will prefer to plot with a dataframe made from value_counts()
sorted(finals_teams.items(), key=lambda x: x[1], reverse=True)
```

Out[18]:
```
[('Italy', 6),
 ('Germany FR', 6),
 ('Brazil', 6),
 ('Argentina', 5),
 ('Netherlands', 3),
 ('Germany', 2),
 ('Czechoslovakia', 2),
 ('Hungary', 2),
 ('France', 2),
 ('Uruguay', 1),
 ('England', 1),
 ('Sweden', 1),
 ('Spain', 1)]
```

In [19]:

```
#Make a dataframe from counting values in all_finals_df
finals_teams_ranked = all_finals_df.Teams.value_counts()
finals_teams_ranked_df = pd.DataFrame(finals_teams_ranked)
finals_teams_ranked_df.reset_index(inplace=True)
finals_teams_ranked_df.columns = ['Teams', 'Frequency'] #rename the columns as neea

fig = px.bar(finals_teams_ranked_df, x='Teams', y='Frequency', color='Frequency', h
             labels={'Teams':'Teams in the Finals'},
            title="All Teams Who Have Reached Finals and Frequency")
fig.update_layout(uniformtext_minsize=8)
fig.show()
```



Que 4: All teams who have reached semis and how many times.

- Much like the previous item but this time done on Semi-finals

In [20]:
```
#Take a piece of the df corresponding to all 'Semi-finals' in the 'Round' column
all_semi_finals = df[df['Round'] == 'Semi-finals']
all_semi_finals.head()
```

Out[20]:

	Year	month	day	Time	Round	HomeTeam	HomeGoals	AwayTeam	AwayGoals	TotalGoals
15	1930	Jul	Friday	14:45	Semi-finals	Argentina	6	USA	1	
16	1930	Jul	Saturday	14:45	Semi-finals	Uruguay	6	Yugoslavia	1	
31	1934	Jun	Saturday	16:30	Semi-finals	Italy	1	Austria	0	
32	1934	Jun	Saturday	16:30	Semi-finals	Czechoslovakia	3	Germany	1	
49	1938	Jun	Wednesday	18:00	Semi-finals	Hungary	5	Sweden	1	

In [21]:
```
#Make a list of all teams invovled Home and Away
teams = [line for line in all_semi_finals.HomeTeam] + [line for line in all_semi_fi
```

In [22]:
```
#Make a df of teams
all_semi_finals_df = pd.DataFrame(columns=['Teams'], data = teams)
#peep the head()
all_semi_finals_df.head(2)
```

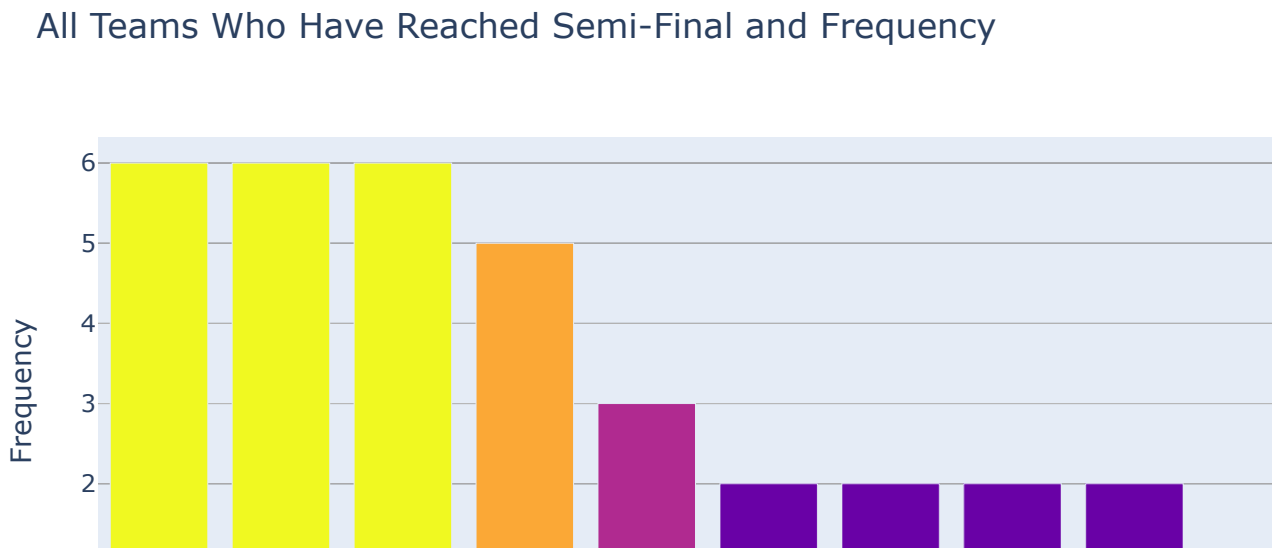
Out[22]:

	Teams
0	Argentina
1	Uruguay

In [23]:

```
#Make a dataframe from counting values in all_semi_finals
#See 'Finals' cell above for explanation as the steps are identical. We are avoidin
semifinals_teams_ranked = all_semi_finals_df.Teams.value_counts()
semifinals_teams_ranked_df = pd.DataFrame(semifinals_teams_ranked)
semifinals_teams_ranked_df.reset_index(inplace=True)
semifinals_teams_ranked_df.columns = ['Teams', 'Frequency']

fig = px.bar(finals_teams_ranked_df, x='Teams', y='Frequency', color='Frequency', h
             labels={'Teams':'Teams in the Finals'},
            title="All Teams Who Have Reached Semi-Final and Frequency")
fig.update_layout(uniformtext_minsize=8)
fig.show()
```



Que 5: How many goals and average goals scored in all semi-finals.

In [24]:

```
#Continue with our dataframe holding only semi-final matches
#Let's take a sum and mean of 'TotalGoals'
semi_goals_sum = all_semi_finals.TotalGoals.sum()
semi_goals_ave = all_semi_finals.TotalGoals.mean()
print(f"{semi_goals_sum} goals were scored in all Semi-Finals\nAn average of {semi_
```

```
123 goals were scored in all Semi-Finals
An average of 3.62 in every match.
```

Que 6: How many goals and average goals scored in all quarter-finals.

In [25]:
```
#Get a slice of the original df for all quarter-finals
all_qtrs = df[df['Round'] == 'Quarter-finals']
all_qtrs.head()
```

Out[25]:

	Year	month	day	Time	Round	HomeTeam	HomeGoals	AwayTeam	AwayGoals	TotalGoa
26	1934	May	Wednesday	16:30	Quarter-finals	Czechoslovakia	3	Switzerland	2	
27	1934	May	Wednesday	16:30	Quarter-finals	Germany	2	Sweden	1	
28	1934	May	Wednesday	16:30	Quarter-finals	Italy	1	Spain	1	
29	1934	May	Wednesday	16:30	Quarter-finals	Austria	2	Hungary	1	
30	1934	Jun	Thursday	16:30	Quarter-finals	Italy	1	Spain	0	

In [26]:
```
qtrs_goals_sum = all_qtrs.TotalGoals.sum()
qtrs_goals_ave = all_qtrs.TotalGoals.mean()
print(f"{qtrs_goals_sum} goals were scored in all Semi-Finals\nAn average of {qtrs_
```

```
175 goals were scored in all Semi-Finals
An average of 2.82 in every match.
```

In [27]:
```
#All goals in finals
finals = df[df['Round'] == 'Final']['HomeGoals'].sum() + df[df['Round'] == 'Final']
print(f"{finals} goals in finals")
```

```
68 goals in finals
```

Que 7: How many goals and average number scored in all finals

In [28]:
```
#Working with the piece of the original df holding finals
finals_goals_sum = all_finals.TotalGoals.sum()
finals_goals_ave = all_finals.TotalGoals.mean()
print(f"{finals_goals_sum} goals were scored in all Semi-Finals\nAn average of {fin
```

```
68 goals were scored in all Semi-Finals
An average of 3.58 in every match.
```

Que 8: How many matches were played outside quarter-finals and above.

In [29]:
```
#Let's be creative here!
#First get a slice with no finals
df_less_finals = df[df['Round'] != 'Final']

#From there get a slies with no semi-finals and viola we are left with all matches
df_less_finals_semis = df_less_finals[df_less_finals['Round'] != 'Semi-Finals']

#One more dropping qtrs. This is fun.
df_less_finals_semis_qtrs = df_less_finals_semis[df_less_finals_semis['Round'] != '
```

In [30]:
```
#Did it work? Well, let's check!
'Final' in df_less_finals_semis_qtrs.Round.tolist() or 'Semi-Finals' in df_less_fin
```

Out[30]: False

In [31]:
```
#Just in case that was lady-Luck, let's make sure other Rounds are there
'Round of 16' in df_less_finals_semis_qtrs.Round.tolist()
```

Out[31]: True

In [32]:
```
#Total matches in this slice of the dataframe is same number of rows. A number of m
d_rest0 = df_less_finals_semis_qtrs.shape[0]
d_rest1 = len(df_less_finals_semis_qtrs)

d_rest0 == d_rest1
```

Out[32]: True

In [33]:
```
print(f"There are {d_rest0} matches played outside Quater-finals and above")
```

```
There are 755 matches played outside Quater-finals and above
```

Next: The *Kicker* and the *Teaser Plot*

- Stay Tuned.