FIFA Winner Prediction

Project by:

- Harish Agarwal(200031101611025)
- Jainam Shah(200031101611046)

In [54]:
```
Image(filename='logo.jpg')
```

Out[54]:

Context:

The FIFA World Cup is an international soccer tournament held every four years. The 2022 FIFA World Cup is scheduled to take place in Qatar from November 21 to December 18, 2022. This will be the first time that the World Cup will be held in the Middle East, and the first time that it will take place in November and December, rather than June and July. A total of 32 teams will participate in the tournament.

Objective:

The main objective is to predict the winner of 2022 FIFA world cup with the help of data of all the previous matches played till now.

Data Description

The data contains the different data related to a previous FIFA world cup. The detailed data dictionary is given below.

Data Dictionary:

date:The actual date when the match was played.

home_team:home team is the team which playes in its own country.

away_team:away team is the team which playes matches in any country except its country.

home_score:Score made by home team.

away_score:score made by away team

tournament:There are different types of tournament like FIFA, friendly, British tournament etc.

city:The city in which the match was played.

country: Country in which the match was played.

neutral: Result of the match.

Let us start by importing the required libraries

In [55]:
```
#Importing required Libraries

import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
import random
from IPython.display import Image
random_num=122
```

Understanding the structure of the data

In [56]:
```
#Reading the dataset

df=pd.read_csv("results.csv")
df.head(100)
```

Out[56]:

	date	home_team	away_team	home_score	away_score	tournament	city	country	neutral
0	1872-11-30	Scotland	England	0	0	Friendly	Glasgow	Scotland	False
1	1873-03-08	England	Scotland	4	2	Friendly	London	England	False
2	1874-03-07	Scotland	England	2	1	Friendly	Glasgow	Scotland	False
3	1875-03-06	England	Scotland	2	2	Friendly	London	England	False
4	1876-03-04	Scotland	England	3	0	Friendly	Glasgow	Scotland	False
...	...	...	...	...	...	...	...	...	...
95	1894-03-31	Northern Ireland	Scotland	1	2	British Championship	Belfast	Republic of Ireland	False
96	1894-04-07	Scotland	England	2	2	British Championship	Glasgow	Scotland	False
97	1895-03-09	England	Northern Ireland	9	0	British Championship	Derby	England	False
98	1895-03-16	Northern Ireland	Wales	2	2	British Championship	Belfast	Republic of Ireland	False
99	1895-03-18	England	Wales	1	1	British Championship	London	England	False

100 rows × 9 columns

Observations:

The DataFrame has 9 columns as mentioned in the Data Dictionary. Data in each row corresponds to the matche played by the teams.

Question 1: How many rows and columns are present in the data?

In [57]:
```
print('the data frame has',df.shape[0],'rows and',df.shape[1],'columns')
```

```
the data frame has 42082 rows and 9 columns
```

Observations:

We have 42082 rows and 9 columns in the data. The rows account for all the orders processed

Question 2: What are the datatypes of the different columns in the dataset?

In [58]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42082 entries, 0 to 42081
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   date        42082 non-null  object
 1   home_team   42082 non-null  object
 2   away_team   42082 non-null  object
 3   home_score  42082 non-null  int64 
 4   away_score  42082 non-null  int64 
 5   tournament  42082 non-null  object
 6   city        42082 non-null  object
 7   country     42082 non-null  object
 8   neutral     42082 non-null  bool  
dtypes: bool(1), int64(2), object(6)
memory usage: 2.6+ MB
```

Observations:

We have 1 bool data type, 2 integers and 6 objects.

Question 3: Check the statistical summary of the data. What is the minimum, average, and maximum score of home and away team?

In [59]:
```
df.describe()
```

Out[59]:

	home_score	away_score
count	42082.000000	42082.000000
mean	1.743691	1.186541
std	1.752459	1.403957
min	0.000000	0.000000
25%	1.000000	0.000000
50%	1.000000	1.000000
75%	2.000000	2.000000
max	31.000000	21.000000

Observations:

- we have total 42082 observations.
- mean, std, min and max score of home team is 1.743691, 1.752459, 0.000000 and 31.00000 respectively.
- mean, std, min and max score of away team is 1.186541, 1.403957, 0.000000 and 21.00000 respectively.

Question 4: Are there any missing values in the data? If yes, treat them using an appropriate method?

In [60]:
```
#Preprocessing
#Finding outmissing values

df.isnull().sum()
#No any missing values
#We are good to go
```

Out[60]:
```
date          0
home_team     0
away_team     0
home_score    0
away_score    0
tournament    0
city          0
country       0
neutral       0
dtype: int64
```

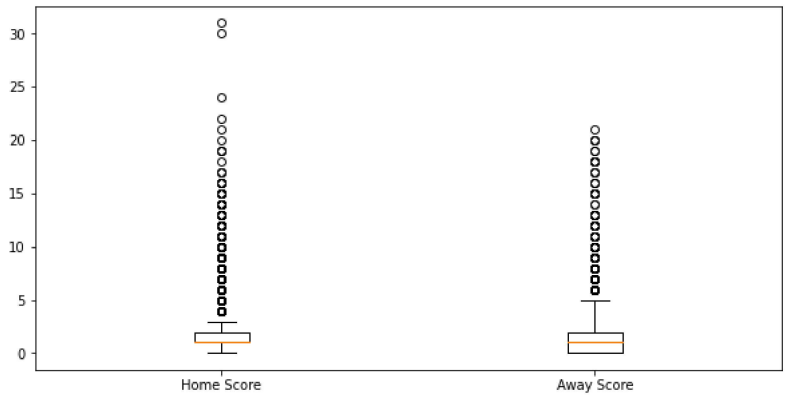
Observations:

We do not have any null values in the data frame, hence it requires no treatment

Question 5: Find out all the outliers in the data columns?

In [61]:
```
fig = plt.figure(figsize =(10, 5))
plt.boxplot([df["home_score"],df["away_score"]])
plt.xticks([1,2], ["Home Score", "Away Score"])
```

Out[61]:
```
([<matplotlib.axis.XTick at 0x7f5ac44bd1c0>,
  <matplotlib.axis.XTick at 0x7f5ac44bdf40>],
 [Text(0, 0, 'Home Score'), Text(0, 0, 'Away Score')])
```



Observations:

- We have created a box plot to find out the outliers
- Boxplot says there are many outliers. Let's remove those outliers.

- It will make our machine learning model more accurate.

Question 6: Remove all the outliers?

In [62]:
```
higher_home=15
higher_away=15
df=df[(df["home_score"]<higher_home) & (df["away_score"]<higher_away)]
df.head(5)
```

Out[62]:

	date	home_team	away_team	home_score	away_score	tournament	city	country	neutral
0	1872-11-30	Scotland	England	0	0	Friendly	Glasgow	Scotland	False
1	1873-03-08	England	Scotland	4	2	Friendly	London	England	False
2	1874-03-07	Scotland	England	2	1	Friendly	Glasgow	Scotland	False
3	1875-03-06	England	Scotland	2	2	Friendly	London	England	False
4	1876-03-04	Scotland	England	3	0	Friendly	Glasgow	Scotland	False

Observations:

- To remove all the outliers we assume that the maximum goals that one team can score is 15
- datasets after removing outliers

Question 7: Make a new column to represent the result of all the matches?

In [63]:
```
conditions  = [df["home_score"] ==df["away_score"], df["home_score"] > df["away_score"] , df["home_score"] < df["away_score"]]
choices     = [ "Draw", 'Win', 'Lost' ]
df["Win_Statues"] = np.select(conditions, choices)
df.head(5)
```

Out[63]:

	date	home_team	away_team	home_score	away_score	tournament	city	country	neutral	Win_Statues
0	1872-11-30	Scotland	England	0	0	Friendly	Glasgow	Scotland	False	Draw
1	1873-03-08	England	Scotland	4	2	Friendly	London	England	False	Win
2	1874-03-07	Scotland	England	2	1	Friendly	Glasgow	Scotland	False	Win
3	1875-03-06	England	Scotland	2	2	Friendly	London	England	False	Draw
4	1876-03-04	Scotland	England	3	0	Friendly	Glasgow	Scotland	False	Win

Observations:

- Making a new column named "Win_Statues"
- "Win_Statues" will store the result(Win, Lost, Draw) of the home team

Question 8: List down all the countries present in the dataset?

In [64]:
```
countries=df.home_team.unique()
print(f"There are {len(countries)} Countries in the home_team Column\n")
print(f"Countries-{countries}")
```

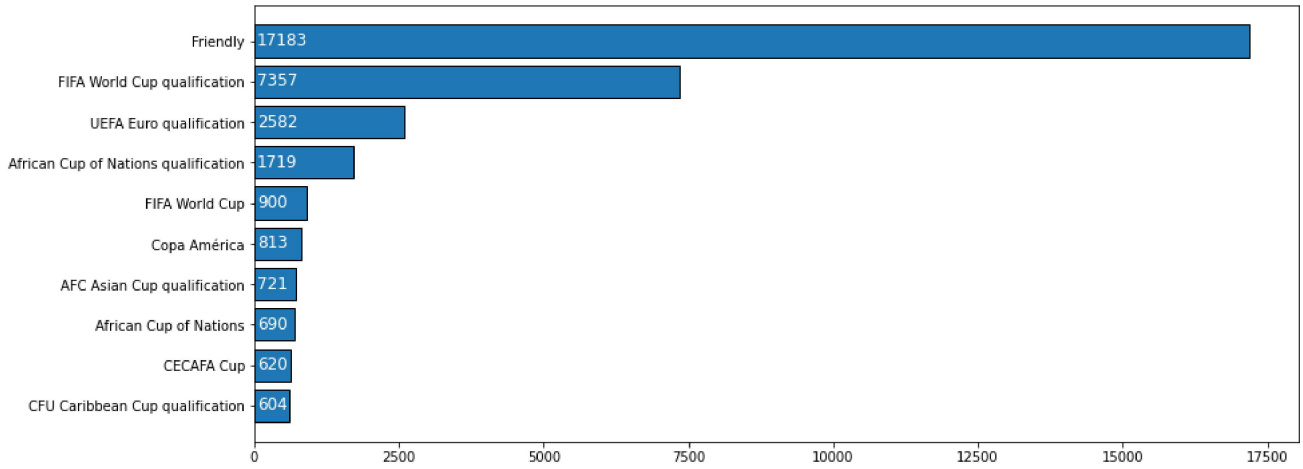
```
There are 306 Countries in the home_team Column

Countries-['Scotland' 'England' 'Wales' 'Northern Ireland' 'United States' 'Uruguay'
 'Austria' 'Hungary' 'Argentina' 'Belgium' 'France' 'Netherlands'
 'Czechoslovakia' 'Switzerland' 'Sweden' 'Germany' 'Italy' 'Chile'
 'Norway' 'Finland' 'Luxembourg' 'Russia' 'Denmark' 'Catalonia'
 'Basque Country' 'Brazil' 'Paraguay' 'Japan' 'Canada' 'Estonia'
 'Costa Rica' 'Guatemala' 'Spain' 'Brittany' 'Poland' 'Yugoslavia'
 'New Zealand' 'Romania' 'Latvia' 'Galicia' 'Portugal' 'Andalusia'
 'China PR' 'Australia' 'Lithuania' 'Turkey' 'Central Spain' 'Mexico'
 'Aruba' 'Egypt' 'Haiti' 'Philippines' 'Bulgaria' 'Jamaica' 'Kenya'
 'Bolivia' 'Peru' 'Honduras' 'Guyana' 'Uganda' 'Belarus' 'El Salvador'
 'Barbados' 'Republic of Ireland' 'Trinidad and Tobago' 'Greece' 'Curaçao'
 'Dominica' 'Silesia' 'Guadeloupe' 'Israel' 'Suriname' 'French Guiana'
 'Cuba' 'Colombia' 'Ecuador' 'Saint Kitts and Nevis' 'Panama' 'Slovakia'
 'Manchukuo' 'Croatia' 'Nicaragua' 'Afghanistan' 'India' 'Martinique'
 'Zimbabwe' 'Iceland' 'Albania' 'Madagascar' 'Zambia' 'Mauritius'
 'Tanzania' 'Iran' 'Djibouti' 'DR Congo' 'Vietnam' 'Macau' 'Ethiopia'
 'Puerto Rico' 'Réunion' 'Sierra Leone' 'Zanzibar' 'South Korea' 'Ghana'
 'South Africa' 'New Caledonia' 'Fiji' 'Nigeria' 'Venezuela' 'Burma'
 'Sri Lanka' 'Tahiti' 'Gambia' 'Hong Kong' 'Singapore' 'Malaysia'
 'Indonesia' 'Guinea-Bissau' 'German DR' 'Vanuatu' 'Kernow' 'Saarland'
 'Cambodia' 'Lebanon' 'Pakistan' 'Vietnam Republic' 'North Korea' 'Togo'
 'Sudan' 'Malta' 'Syria' 'Tunisia' 'Malawi' 'Morocco' 'Benin' 'Cameroon'
 'Central African Republic' 'Gabon' 'Ivory Coast' 'Congo' 'Mali'
 'North Vietnam' 'Mongolia' 'Chinese Taipei' 'Cyprus' 'Iraq' 'Saint Lucia'
 'Grenada' 'Thailand' 'Senegal' 'Libya' 'Guinea' 'Algeria' 'Kuwait'
 'Jordan' 'Solomon Islands' 'Liberia' 'Laos'
 'Saint Vincent and the Grenadines' 'Bermuda' 'Niger' 'Bahrain'
 'Montenegro' 'Palestine' 'Papua New Guinea' 'Burkina Faso' 'Mauritania'
 'Saudi Arabia' 'Eswatini' 'Somalia' 'Lesotho' 'Qatar'
 'Antigua and Barbuda' 'Faroe Islands' 'Bangladesh' 'Oman' 'Yemen DPR'
 'Burundi' 'Yemen' 'Mozambique' 'Guam' 'Chad' 'Angola'
 'Dominican Republic' 'Seychelles' 'Rwanda' 'São Tomé and Príncipe'
 'Botswana' 'Northern Cyprus' 'Cape Verde' 'Kyrgyzstan' 'Georgia'
 'Azerbaijan' 'Kiribati' 'Tonga' 'Wallis Islands and Futuna'
 'United Arab Emirates' 'Brunei' 'Equatorial Guinea' 'Liechtenstein'
 'Nepal' 'Greenland' 'Niue' 'Samoa' 'American Samoa' 'Belize' 'Anguilla'
 'Cayman Islands' 'Palau' 'Sint Maarten' 'Namibia' 'Åland Islands'
 'Ynys Môn' 'Saint Martin' 'San Marino' 'Slovenia' 'Jersey' 'Shetland'
 'Isle of Wight' 'Moldova' 'Ukraine' 'Kazakhstan' 'Tajikistan'
 'Uzbekistan' 'Turkmenistan' 'Armenia' 'Czech Republic' 'Guernsey'
 'Gibraltar' 'Isle of Man' 'North Macedonia' 'Montserrat' 'Serbia'
 'Cook Islands' 'Canary Islands' 'Bosnia and Herzegovina' 'Maldives'
 'Andorra' 'British Virgin Islands' 'Frøya' 'Hitra' 'U.S. Virgin Islands'
 'Corsica' 'Eritrea' 'Bahamas' 'Gotland' 'Saare County' 'Rhodes'
 'Micronesia' 'Bhutan' 'Orkney' 'Monaco' 'Tuvalu' 'Alderney' 'Mayotte'
 'Turks and Caicos Islands' 'East Timor' 'Western Isles'
 'Falkland Islands' 'Kosovo' 'Republic of St. Pauli' 'Găgăuzia' 'Tibet'
 'Crimea' 'Occitania' 'Sápmi' 'Northern Mariana Islands' 'Menorca'
 'Comoros' 'Provence' 'Arameans Suryoye' 'Padania' 'Iraqi Kurdistan'
 'Gozo' 'Bonaire' 'Western Sahara' 'Raetia' 'Tamil Eelam' 'South Sudan'
 'Abkhazia' 'Artsakh' 'Madrid' 'Vatican City' 'Ellan Vannin'
 'South Ossetia' 'Darfur' 'County of Nice' 'Székely Land' 'Romani people'
 'Felvidék' 'Chagos Islands' 'United Koreans in Japan' 'Somaliland'
 'Western Armenia' 'Barawa' 'Kárpátalja' 'Yorkshire' 'Panjab'
 'Matabeleland' 'Cascadia' 'Kabylia' 'Timor-Leste' 'Myanmar'
 'Parishes of Jersey' 'Chameria' 'Saint Helena']
```

Observations:

- all the countries present in the dataset are listed above

Question 9: Visually represent all the types of tournament ever played?

In [65]:
```
rank_bound = 10
ax = df.tournament.value_counts()[:rank_bound].sort_values()
value = ax.values
label = ax.index

plt.figure(figsize=(14,6))
plt.barh(y=label, width=value, edgecolor="k")
for i in range(rank_bound):
    plt.text(x=50,y=i-0.1,s=value[i],color="w",fontsize=12)
plt.show()
```

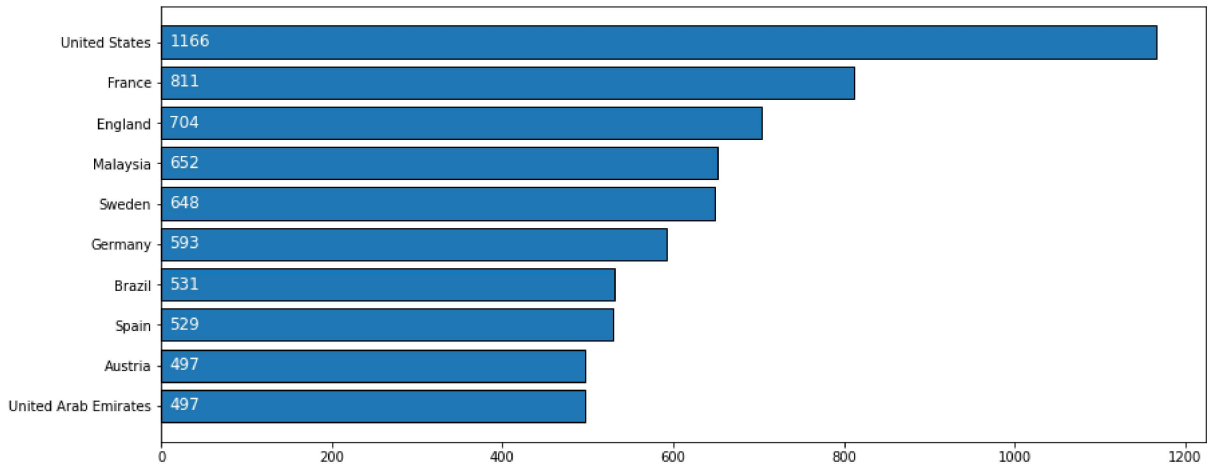


Observations:

There are total 10 types of tournament ever played and they are represented virtually above.

Question 10: Which country has played most of the matches and which country has played minimum matches?

In [66]:
```
rank_bound = 10
ax = df.country.value_counts()[:rank_bound].sort_values()
value = ax.values
label = ax.index

plt.figure(figsize=(14,6))
plt.barh(y=label, width=value, edgecolor="k")
for i in range(rank_bound):
    plt.text(x=10,y=i-0.1,s=value[i],color="w",fontsize=12)
plt.show()
```

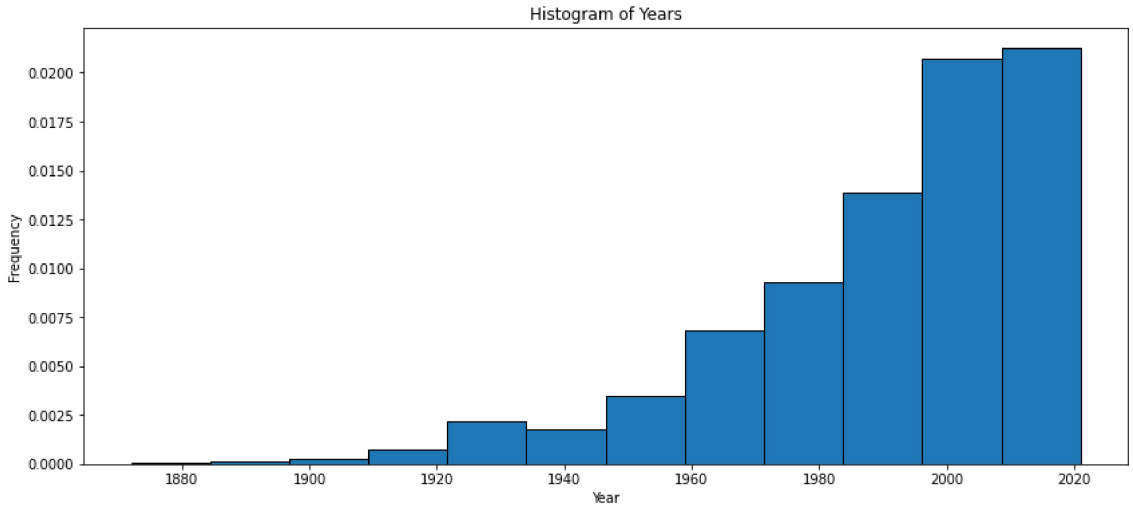


Observations:

- To find out the the which country has played maximum and minimum matches, we have plot a bar plot.
- From the above bar plot, it is clear that United States has played most of the matches and Austria has played minimum matches.

Question 11: Which year in the history can be called as the start of golden era of football history?

In [67]:
```
years = []
for date in df.date:
    years.append(int(str(date)[0:4]))
plt.figure(figsize=(14,6))
plt.hist(years, density=True, bins=12, edgecolor="k")
plt.title("Histogram of Years")
plt.ylabel("Frequency")
plt.xlabel("Year")
plt.show()
```



Observations:

- Most matches are played after 1960.
- Hence we can say that 1960 is the start of golden era of football history.

Question 12: Write a code to remove all the friendly tournament matches.

In [68]:
```
data_home=df.loc[df["home_team"]==df["country"] ]
data_home=df.loc[df["tournament"] != "Friendly"]
data_home.head(5)
```

Out[68]:

	date	home_team	away_team	home_score	away_score	tournament	city	country	neutral	Win_Statues
29	1884-01-26	Northern Ireland	Scotland	0	5	British Championship	Belfast	Republic of Ireland	False	Lost
30	1884-02-09	Wales	Northern Ireland	6	0	British Championship	Wrexham	Wales	False	Win
31	1884-02-23	Northern Ireland	England	1	8	British Championship	Belfast	Republic of Ireland	False	Lost
32	1884-03-15	Scotland	England	1	0	British Championship	Glasgow	Scotland	False	Win
33	1884-03-17	Wales	England	0	4	British Championship	Wrexham	Wales	False	Lost

Observations:

- In order to predict the winner we have to remove friendly tournament matches.
- we will take home_team == Country and remove the data of friendly matches.

Question 13: Describe the performance of the home team in tournament matches.

In [69]:
```
tournament_df=pd.crosstab(data_home["tournament"], data_home["Win_Statues"],margins=True)
tournament_df=tournament_df.sort_values("All",ascending=False).head(10)
tournament_df.style.bar(color="brown",subset=["Draw","Lost","Win","All"])
```

Out[69]:

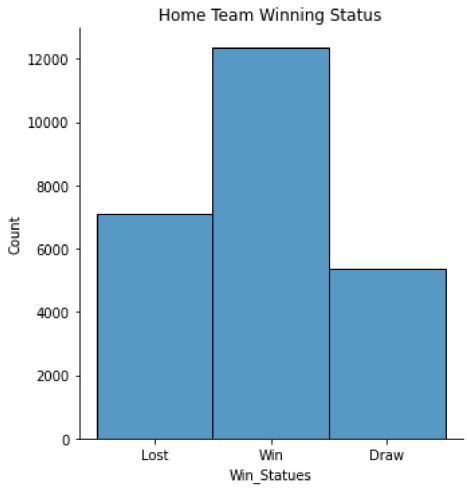
Win_Statues / tournament	Draw	Lost	Win	All
All	5388	7111	12350	24849
FIFA World Cup qualification	1570	2000	3787	7357
UEFA Euro qualification	506	815	1261	2582
African Cup of Nations qualification	418	353	948	1719
FIFA World Cup	199	290	411	900
Copa América	172	225	416	813
AFC Asian Cup qualification	123	174	424	721
African Cup of Nations	193	186	311	690
CECAFA Cup	147	185	288	620
CFU Caribbean Cup qualification	124	170	310	604

Observation:

above table shows the data of draw, lost, win matches of home team in all the tournaments.

In [70]:
```
sns.displot(data_home, x="Win_Statues")
plt.title("Home Team Winning Status")
```

Out[70]:
```
Text(0.5, 1.0, 'Home Team Winning Status')
```



Observation:

Displot shows the result of home team visually but we want accurate data so that we can get more accuracy.

In [71]:
```
data_home['Win_Statues'].value_counts()
```

Out[71]:
```
Win     12350
Lost     7111
Draw     5388
Name: Win_Statues, dtype: int64
```

Observations:

- Total matches won by the home team is 12350.
- Total matches Lost by the home team is 7111.
- Total matches draw by the home team is 5388.
- This is the overall performance of the home team.

Question 14: How each team perform while playing a tournament in home grounds?

In [72]:
```
teams_win_statues=pd.crosstab(df["home_team"], df["Win_Statues"],margins=True, margins_name="Total")
teams_win_statues["team_win_probability"]=teams_win_statues["Win"]/(teams_win_statues["Total"])
teams_win_statues_100=teams_win_statues.loc[teams_win_statues["Total"]>200]
teams_win_statues_100=teams_win_statues_100.sort_values("team_win_probability",ascending=False)
teams_win_statues_100.head(20).style.bar(color="orange",subset="team_win_probability")
```

Out[72]:

Win_Statues	Draw	Lost	Win	Total	team_win_probability
home_team					
Brazil	108	56	406	570	0.712281
Spain	66	49	249	364	0.684066
Argentina	120	68	362	550	0.658182
Ivory Coast	64	35	175	274	0.638686
Iran	61	42	177	280	0.632143
Nigeria	67	33	171	271	0.630996
Egypt	68	74	239	381	0.627297
Italy	119	49	281	449	0.625835
Germany	109	84	320	513	0.623782
England	109	82	314	505	0.621782
Ghana	71	39	177	287	0.616725
Russia	70	49	185	304	0.608553
Czechoslovakia	51	40	139	230	0.604348
Morocco	68	47	171	286	0.597902
South Korea	112	79	277	468	0.591880
Mexico	114	98	303	515	0.588350
Senegal	55	36	130	221	0.588235
Costa Rica	77	50	179	306	0.584967
Iraq	71	44	160	275	0.581818
Sweden	105	101	286	492	0.581301

Observations:

- Lets take teams which plays atleast 200 games
- We can see teams has more than 50% wining probability while playing in the home grounds

Question 15:How each team perform while playing a tournament in away grounds?

In [73]:
```
teams_away_statues=pd.crosstab(df["away_team"], df["Win_Statues"],margins=True, margins_name="Total")
teams_away_statues["team_win_probability"]=teams_away_statues["Lost"]/(teams_away_statues["Total"])
teams_away_statues_100=teams_away_statues.loc[teams_away_statues["Total"]>200]
teams_away_statues_100=teams_away_statues_100.sort_values("team_win_probability",ascending=False)
teams_away_statues_100.rename(columns={'Lost': 'Win'}, index={'Win': 'Lost'}, inplace=True)
teams_away_statues_100.head(20)
```

Out[73]:

Win_Statues	Draw	Win	Win	Total	team_win_probability
away_team					
Brazil	91	223	101	415	0.537349
Germany	89	240	119	448	0.535714
England	134	266	115	515	0.516505
Spain	97	162	82	341	0.475073
South Korea	110	177	109	396	0.446970
Netherlands	81	163	129	373	0.436997
Russia	115	175	111	401	0.436409
Iran	67	96	57	220	0.436364
Japan	53	111	93	257	0.431907
Italy	108	150	105	363	0.413223
Australia	56	92	76	224	0.410714
Sweden	120	220	198	538	0.408922
Yugoslavia	61	115	115	291	0.395189
France	80	142	139	361	0.393352
Zambia	103	165	153	421	0.391924
Mexico	87	140	133	360	0.388889
Portugal	63	114	117	294	0.387755
Scotland	87	157	161	405	0.387654
Argentina	124	167	143	434	0.384793
Hungary	105	183	195	483	0.378882

Observations:

- Lets take teams which plays atleast 200 games
- The probability to wining a tournament in away conditions is very low comapred to the winning probalities in home conditions

Question 16: Convert the data set into 10 terms and findsout who has done best at each term? Note: Take home conditions

In [74]:
```
range_years=max(years)-min(years)
no_0f_terms=10
term_size=int(range_years/no_0f_terms)
for i in range(no_0f_terms+1):
    start=years.index(term_size*i+min(years))
    end=years.index(min(term_size*(i+1)+min(years),2021))
    term=df.iloc[start:end]
    best_teams=pd.crosstab(term["home_team"], term["Win_Statues"],margins=True, margins_name="Total")
    best_teams["team_win_probability"]=best_teams["Win"]/(best_teams["Total"])
    best_teams=best_teams.sort_values("team_win_probability",ascending=False)
    best_teams=best_teams.loc[best_teams["Total"]>20]
    if (best_teams.shape[0]>2):
        print(f"\nBest 2 team in the term: {term_size*i+min(years)}-{min(term_size*(i+1)+min(years),2021)} ")
        print(best_teams.iloc[0:2].to_markdown())
    else:
        print(f"No Enough data to find the best team in the term: {term_size*i+min(years)}-{min(term_size*(i+1)+min(years),2021)}
```

No Enough data to find the best team in the term: 1872-1886

Best 2 team in the term: 1886-1900

home_team	Draw	Lost	Win	Total	team_win_probability
England	1	3	18	22	0.818182
Scotland	3	3	16	22	0.727273

Best 2 team in the term: 1900-1914

home_team	Draw	Lost	Win	Total	team_win_probability
England	6	1	14	21	0.666667
Hungary	6	3	16	25	0.64

Best 2 team in the term: 1914-1928

home_team	Draw	Lost	Win	Total	team_win_probability
Denmark	4	3	19	26	0.730769
Argentina	10	9	39	58	0.672414

Best 2 team in the term: 1928-1942

home_team	Draw	Lost	Win	Total	team_win_probability
Argentina	3	4	34	41	0.829268
England	1	4	20	25	0.8

Best 2 team in the term: 1942-1956

home_team	Draw	Lost	Win	Total	team_win_probability
Costa Rica	0	2	20	22	0.909091
Hungary	5	2	34	41	0.829268

Best 2 team in the term: 1956-1970

home_team	Draw	Lost	Win	Total	team_win_probability
China PR	2	3	18	23	0.782609
Ghana	10	0	30	40	0.75

Best 2 team in the term: 1970-1984

home_team	Draw	Lost	Win	Total	team_win_probability
Russia	7	7	39	53	0.735849
Brazil	22	4	68	94	0.723404

Best 2 team in the term: 1984-1998

home_team	Draw	Lost	Win	Total	team_win_probability
Czech Republic	2	3	20	25	0.8
Nigeria	9	3	37	49	0.755102

Best 2 team in the term: 1998-2012

home_team	Draw	Lost	Win	Total	team_win_probability
Spain	10	10	77	97	0.793814
Jersey	4	2	19	25	0.76

Best 2 team in the term: 2012-2021

home_team	Draw	Lost	Win	Total	team_win_probability
Egypt	6	8	47	61	0.770492
Spain	8	5	43	56	0.767857

Observations:

- We will take teams which plays atleast 20 games.
- Best 2 team in the term 1886-1900 is England and scotland.

- Best 2 team in the term 1886-1914 is England and Hungary.
- Best 2 team in the term 1914-1928 is Denmark and Argentina.
- Best 2 team in the term 1928-1948 is Argentina and England.
- Best 2 team in the term 1948-1956 is Costa Rica and Hungary.
- Best 2 team in the term 1956-1970 is China PR and Ghana.
- Best 2 team in the term 1970-1984 is Russia and Brazil.
- Best 2 team in the term 1984-1998 is Czech Republic and Nigeria.
- Best 2 team in the term 1998-2012 is Spain and Jersey.
- Best 2 team in the term 2012-2021 is Egypt and Spain.

Question 17: Convert the data set into 10 terms and findsout who has done best at each term? Note: Take away conditions

In [75]:
```
range_years=max(years)-min(years)
no_0f_terms=10
term_size=int(range_years/no_0f_terms)
for i in range(no_0f_terms+1):
    start=years.index(term_size*i+min(years))
    end=years.index(min(term_size*(i+1)+min(years),2021))
    term=df.iloc[start:end]
    best_teams=pd.crosstab(term["away_team"], term["Win_Statues"],margins=True, margins_name="Total")
    best_teams["team_win_probability"]=best_teams["Lost"]/(best_teams["Total"])
    best_teams=best_teams.sort_values("team_win_probability",ascending=False)
    best_teams=best_teams.loc[best_teams["Total"]>20]
    print(f"\nBest 2 team in the term: {term_size*i+min(years)}-{min(term_size*(i+1)+min(years),2021)} ")
    if (best_teams.shape[0]>2):
        print(best_teams.iloc[0:2].to_markdown())
    else:
        print(f"No Enough data to find the best team in the term: {term_size*i+min(years)}-{min(term_size*(i+1)+min(years),2021)}")
```

Best 2 team in the term: 1872-1886
No Enough data to find the best team in the term: 1872-1886

Best 2 team in the term: 1886-1900

away_team	Draw	Lost	Win	Total	team_win_probability
Scotland	4	13	4	21	0.619048
Total	11	30	44	85	0.352941

Best 2 team in the term: 1900-1914

away_team	Draw	Lost	Win	Total	team_win_probability
England	6	18	4	28	0.642857
Total	44	91	123	258	0.352713

Best 2 team in the term: 1914-1928

away_team	Draw	Lost	Win	Total	team_win_probability
England	8	13	5	26	0.5
Uruguay	8	20	17	45	0.444444

Best 2 team in the term: 1928-1942

away_team	Draw	Lost	Win	Total	team_win_probability
Germany	6	32	20	58	0.551724
Scotland	5	16	8	29	0.551724

Best 2 team in the term: 1942-1956

away_team	Draw	Lost	Win	Total	team_win_probability
Hungary	7	31	9	47	0.659574
Yugoslavia	7	24	9	40	0.6

Best 2 team in the term: 1956-1970

away_team	Draw	Lost	Win	Total	team_win_probability
Zambia	3	12	6	21	0.571429
Brazil	15	46	20	81	0.567901

Best 2 team in the term: 1970-1984

away_team	Draw	Lost	Win	Total	team_win_probability
Tahiti	3	21	5	29	0.724138
Germany	15	40	14	69	0.57971

Best 2 team in the term: 1984-1998

away_team	Draw	Lost	Win	Total	team_win_probability
Brazil	18	41	17	76	0.539474
Germany	22	44	17	83	0.53012

Best 2 team in the term: 1998-2012

away_team	Draw	Lost	Win	Total	team_win_probability
Spain	20	50	12	82	0.609756
France	22	46	12	80	0.575

Best 2 team in the term: 2012-2021

away_team	Draw	Lost	Win	Total	team_win_probability
Brazil	7	33	6	46	0.717391
Belgium	4	31	9	44	0.704545

Observations:

- We will take teams which plays atleast 20 games.

- Best 2 team in the term 1886-1900 is scotland.
- Best 2 team in the term 1886-1914 is England.
- Best 2 team in the term 1914-1928 is England and Urguay.
- Best 2 team in the term 1928-1942 is Germany and Scotland.
- Best 2 team in the term 1942-1956 is Yugoslavia and Hungary.
- Best 2 team in the term 1956-1970 is Zambia and Brazil.
- Best 2 team in the term 1970-1984 is Tahiti and Germany.
- Best 2 team in the term 1984-1998 is Brazil and Germany.
- Best 2 team in the term 1998-2012 is Spain and France.
- Best 2 team in the term 2012-2021 is Brazil and Belgium.

In [76]:
```
df_match=df.copy()
df_match.head(8)
```

Out[76]:

	date	home_team	away_team	home_score	away_score	tournament	city	country	neutral	Win_Statues
0	1872-11-30	Scotland	England	0	0	Friendly	Glasgow	Scotland	False	Draw
1	1873-03-08	England	Scotland	4	2	Friendly	London	England	False	Win
2	1874-03-07	Scotland	England	2	1	Friendly	Glasgow	Scotland	False	Win
3	1875-03-06	England	Scotland	2	2	Friendly	London	England	False	Draw
4	1876-03-04	Scotland	England	3	0	Friendly	Glasgow	Scotland	False	Win
5	1876-03-25	Scotland	Wales	4	0	Friendly	Glasgow	Scotland	False	Win
6	1877-03-03	England	Scotland	1	3	Friendly	London	England	False	Lost
7	1877-03-05	Wales	Scotland	0	2	Friendly	Wrexham	Wales	False	Lost

Observation:

We have created a copy of main dataframe for future use.

Making of new dataset with required features to train the machine learning model.

In [77]:
```
#Year,Played Country,Team_1,team_2,team_1 score,team_2 score

New_Dataset_part_1=pd.DataFrame(list(zip(years,df_match.values[:,7],df_match.values[:,1],df_match.values[:,2],df_match.values[:,3
#Making a new dataset by changing the team_1 and team_2 and their respective scores
New_Dataset_part_2=pd.DataFrame(list(zip(years,df_match.values[:,7],df_match.values[:,2],df_match.values[:,1],df_match.values[:,4
New_Dataset=pd.concat([New_Dataset_part_1,New_Dataset_part_2],axis=0)
New_Dataset =New_Dataset.sample(frac=1).reset_index(drop=True) #Shaffling the dataset
New_Dataset.head(5)
```

Out[77]:

	year	Country	team_1	team_2	team_1_score	team_2_score
0	1984	Norway	Yugoslavia	Norway	1	1
1	1997	Senegal	Senegal	Mali	1	2
2	2008	India	India	Chinese Taipei	3	0
3	1984	Singapore	South Korea	Saudi Arabia	1	1
4	1989	United States	United States	Peru	3	0

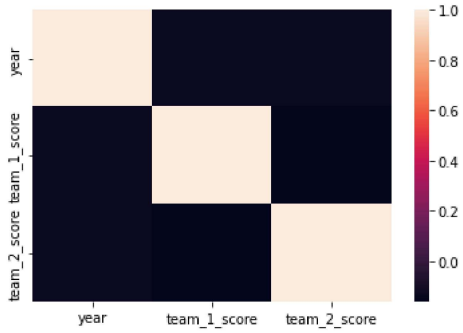
In [78]:
```
#Creating a list containg all the names of the countries

teams_1=New_Dataset.team_1.unique()
contries=New_Dataset.Country.unique()
all_countries=np.unique(np.concatenate((teams_1,contries), axis=0))
len(all_countries)
```

Out[78]: 343

In [79]:
```
#Making a heatmap to see the correlation of each columns

sns.heatmap(New_Dataset.corr())
New_Dataset.corr()
```

Out[79]:

	year	team_1_score	team_2_score
year	1.000000	-0.127807	-0.127807
team_1_score	-0.127807	1.000000	-0.160107
team_2_score	-0.127807	-0.160107	1.000000



In [80]:
```
#Defining the features and labels(Targets)

Y= New_Dataset.iloc[:,4:6] #Training targets (team_1_score and team_2_score)
categorized_data=New_Dataset.iloc[:,0:4].copy() #Traing features

from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()

#Labeling the data using LabelEncorder in Sklearn-(Giving a unique number to each string(country))

label_encoder.fit(all_countries)
#list(label_encoder.classes_)
categorized_data['team_1']=label_encoder.transform(categorized_data['team_1'])
categorized_data['team_2']=label_encoder.transform(categorized_data['team_2'])
categorized_data['Country']=label_encoder.transform(categorized_data['Country'])

#Converting these feature columns to categrize form to make the training processs more smoother
categorized_data['team_1']=categorized_data['team_1'].astype("category")
categorized_data['team_2']=categorized_data['team_2'].astype("category")
categorized_data['Country']=categorized_data['team_2'].astype("category")
```

In [81]:
```
#Input Features to the model (x)

categorized_data.head(5)
```

Out[81]:

	year	Country	team_1	team_2
0	1984	216	336	216
1	1997	176	260	176
2	2008	64	135	64
3	1984	258	275	258
4	1989	230	314	230

In [82]:
```
#Targets to the model (Y)

Y.head(5)
```

Out[82]:

	team_1_score	team_2_score
0	1	1
1	1	2
2	3	0
3	1	1
4	3	0

In [83]:
```
#Info about the X and Y dataframes

print(categorized_data.info())
print(Y.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 84064 entries, 0 to 84063
Data columns (total 4 columns):
 #   Column   Non-Null Count  Dtype   
---  ------   --------------  -----   
 0   year     84064 non-null  int64   
 1   Country  84064 non-null  category
 2   team_1   84064 non-null  category
 3   team_2   84064 non-null  category
dtypes: category(3), int64(1)
memory usage: 1.2 MB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 84064 entries, 0 to 84063
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   team_1_score  84064 non-null  int64
 1   team_2_score  84064 non-null  int64
dtypes: int64(2)
memory usage: 1.3 MB
None
```

Prediction Model

In [84]:
```
#Making the model

X=categorized_data
from sklearn.multioutput import MultiOutputRegressor
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier

model = MultiOutputRegressor(RandomForestClassifier())
model.fit(X,Y)
```

Out[84]:
```
MultiOutputRegressor(estimator=RandomForestClassifier())
```

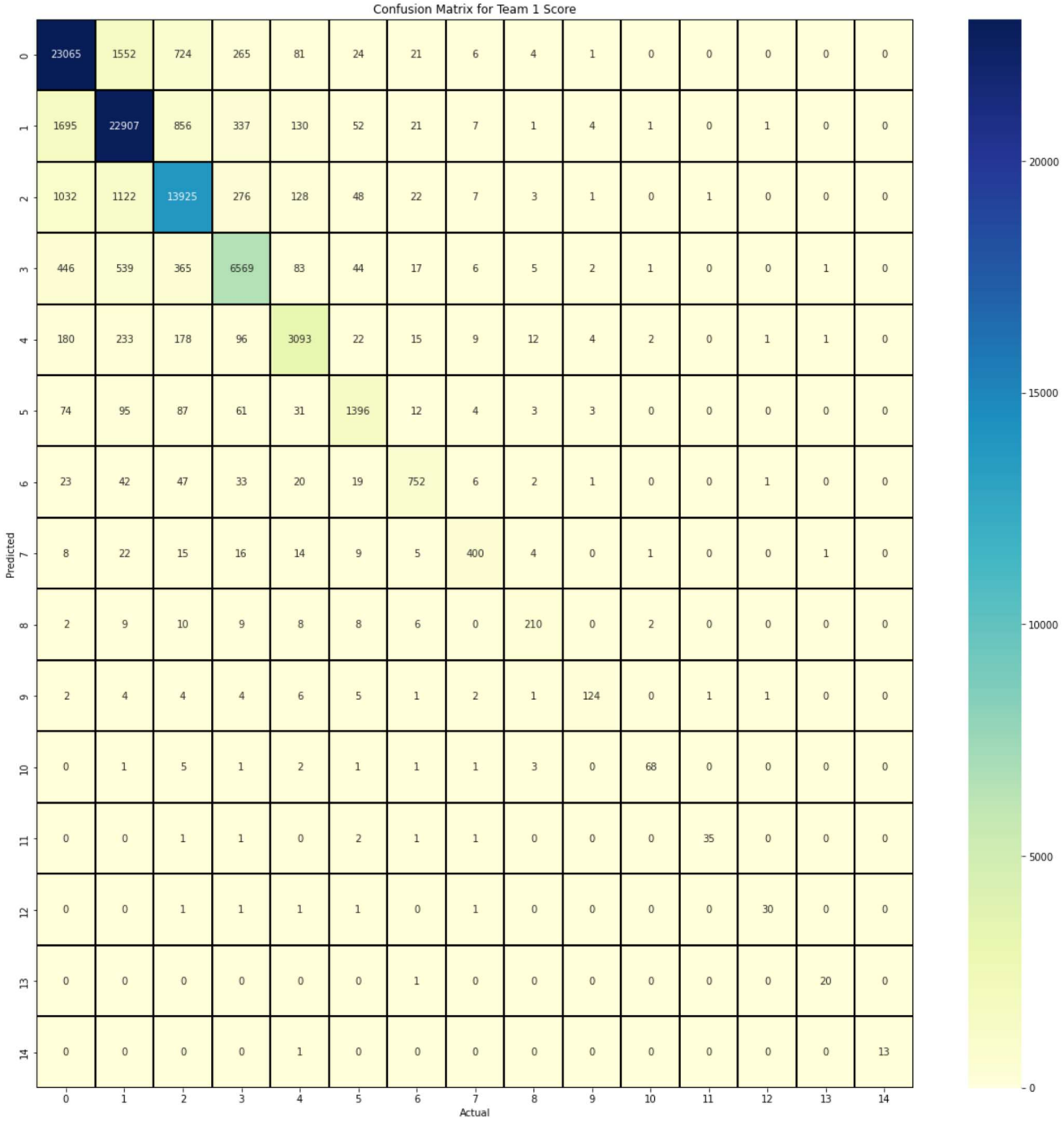
In [85]:
```
#Making the predictions

prd=model.predict(X)
prd
```

Out[85]:
```
array([[1, 1],
       [1, 1],
       [3, 2],
       ...,
       [2, 0],
       [0, 0],
       [1, 4]])
```

In [86]:
```
#Creating the Confusion matrix for each predictions

score_team_1=[i[0] for i in prd]
score_team_2=[i[1] for i in prd]

from sklearn.metrics import confusion_matrix
cm1=confusion_matrix(list(Y.iloc[:,0]),score_team_1)
cm2=confusion_matrix(list(Y.iloc[:,1]),score_team_2)
```

In [87]:
```
#Plotting the Confussion Matrix for score of team 01

plt.figure(figsize=(20,20))
sns.heatmap(cm1, annot=True,fmt="d", cmap='YlGnBu', linecolor='black', linewidths=1)
plt.title("Confusion Matrix for Team 1 Score")
plt.xlabel("Actual")
plt.ylabel("Predicted")
```

Out[87]: Text(159.0, 0.5, 'Predicted')

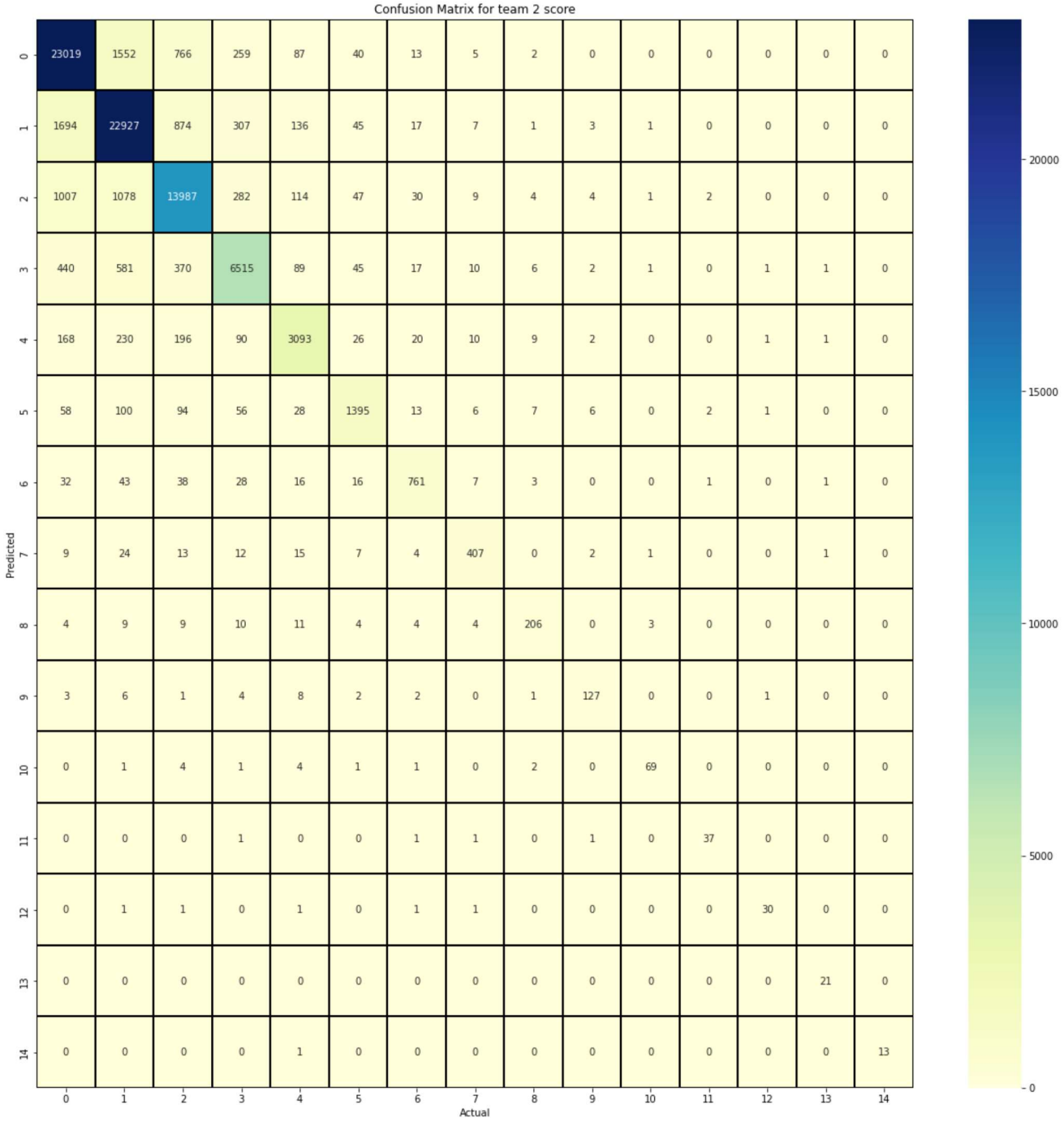


In [88]:
```
#Classification Report to team 1 Score

from sklearn.metrics import classification_report
report_1=classification_report(Y.iloc[:,0],score_team_1)
print(report_1)

#Has a good Accuracy in predicting the team 1 Score
```

```
              precision    recall  f1-score   support

           0       0.87      0.90      0.88     25743
           1       0.86      0.88      0.87     26012
           2       0.86      0.84      0.85     16565
           3       0.86      0.81      0.83      8078
           4       0.86      0.80      0.83      3846
           5       0.86      0.79      0.82      1766
           6       0.86      0.79      0.83       946
           7       0.89      0.81      0.85       495
           8       0.85      0.80      0.82       264
           9       0.89      0.80      0.84       155
          10       0.91      0.82      0.86        83
          11       0.95      0.85      0.90        41
          12       0.88      0.86      0.87        35
          13       0.87      0.95      0.91        21
          14       1.00      0.93      0.96        14

    accuracy                           0.86     84064
   macro avg       0.88      0.84      0.86     84064
weighted avg       0.86      0.86      0.86     84064
```

In [89]:
```
#Plotting the Confussion Matrix for score of team 02

plt.figure(figsize=(20,20))
sns.heatmap(cm2, annot=True,fmt="d", cmap='YlGnBu', linecolor='black', linewidths=1)
plt.title("Confusion Matrix for team 2 score")
plt.xlabel("Actual")
plt.ylabel("Predicted")
```

Out[89]: Text(159.0, 0.5, 'Predicted')



In [90]:
```
#Classification Report to team 2 Score

report_2=classification_report(Y.iloc[:,1],score_team_2)
print(report_2)

#Has a good Accuracy in predicting the team 1 Score#
```

```
              precision    recall  f1-score   support

           0       0.87      0.89      0.88     25743
           1       0.86      0.88      0.87     26012
           2       0.86      0.84      0.85     16565
           3       0.86      0.81      0.83      8078
           4       0.86      0.80      0.83      3846
           5       0.86      0.79      0.82      1766
           6       0.86      0.80      0.83       946
           7       0.87      0.82      0.85       495
           8       0.85      0.78      0.82       264
           9       0.86      0.82      0.84       155
          10       0.91      0.83      0.87        83
          11       0.88      0.90      0.89        41
          12       0.88      0.86      0.87        35
          13       0.84      1.00      0.91        21
          14       1.00      0.93      0.96        14

    accuracy                           0.86     84064
   macro avg       0.88      0.85      0.86     84064
weighted avg       0.86      0.86      0.86     84064
```

In [91]:
```
#Fuction to Select the winning team for the prediction array

def select_winning_team(probability_array):
    prob_lst=[round(probability_array[0][i],3) for i in range(2)]
    if (prob_lst[0]>prob_lst[1]):
        out=0
    elif (prob_lst[0]<prob_lst[1]):
        out=1
    elif (prob_lst[0]==prob_lst[1]):
        out=2
    return out,prob_lst
```

In [92]:
```
#Sample Prediction

mactch_played=2015
team_1="Sri Lanka"
team_2="Brazil"
stadium="Qatar"

team_lst=[team_1,team_2]
team_1_num=label_encoder.transform([team_1])[0]
team_2_num=label_encoder.transform([team_2])[0]
stadium_num=label_encoder.transform([stadium])[0]

print(f"Team 01 is {team_1} -{team_1_num}")
print(f"Team 02 is {team_2} -{team_2_num}")
print(f"Played in  {stadium} -{stadium_num}")
```

```
Team 01 is Sri Lanka -281
Team 02 is Brazil -39
Played in  Qatar -237
```

In [93]:
```
#Sample Prediction Output

X_feature=np.array([[mactch_played,stadium_num,team_1_num,team_2_num]])
res=model.predict(X_feature)
win,_=select_winning_team(res)
try:
    print(f"{team_1} vs {team_2} \n {team_lst[win]} wins 🏆⚽🎯\n")
except IndexError:
    print(f"{team_1} vs {team_2} \n  Match Draw ⚽⚽⚽\n")
```

```
Sri Lanka vs Brazil
 Brazil wins 🏆⚽🎯


/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
```

Observations Sample prediction output given by the prediction Model.

In [94]:
```
Group_A= ["Uruguay","Russia","Saudi Arabia","Egypt"]
Group_B= ["Portugal","Spain","Morocco","Iran"]
Group_C= ["France","Denmark","Australia","Peru"]
Group_D= ["Argentina","Croatia","Iceland","Nigeria"]
Group_E= ["Brazil","Switzerland","Costa Rica","Serbia"]
Group_F= ["South Korea","Mexico","Sweden","Germany"]
Group_G= ["Belgium","England","Panama","Tunisia"]
Group_H= ["Senegal","Japan","Poland","Colombia"]
Groups={"Group A":Group_A,"Group B":Group_B,"Group C":Group_C,"Group D":Group_D,"Group E":Group_E,"Group F":Group_F,"Group G":Gro
```

In [95]:
```
#Initialize basic parameters

year=2022
stadium="Qatar"
stadium_num=label_encoder.transform([stadium])[0]
host_num=stadium_num
```

Q 18. List name of the teams in which qualifies in the Group Stage.

In [96]:
```
##Group stage Matches

Group_standings={}
for grp_name in list(Groups.keys()):
    print(f"{grp_name} Matches")
    probable_countries=Groups[grp_name]
    team_wins_dct={}
    goal_scored_dct={}
    goal_against_dct={}
    win_dct={}
    draw_dct={}
    lost_dct={}
    for i in range(len(probable_countries)):
        j=i+1
        team_1=probable_countries[i]
        team_1_num=label_encoder.transform([team_1])[0]
        team_wins=0
        while j<len((probable_countries)):
            team_2=probable_countries[j]
            team_2_num=label_encoder.transform([team_2])[0]
            team_lst=[team_1,team_2]
            Input_vector=np.array([[year,stadium_num,team_1_num,team_2_num]])
            res=model.predict(Input_vector)

            win,prob_lst=select_winning_team(res)
            goal_scored_dct[team_1] = goal_scored_dct.get(team_1,0)+prob_lst[0]
            goal_scored_dct[team_2] = goal_scored_dct.get(team_2,0)+prob_lst[1]

            goal_against_dct[team_1] = goal_against_dct.get(team_1,0)+prob_lst[1]
            goal_against_dct[team_2] = goal_against_dct.get(team_2,0)+prob_lst[0]

            try:
                print(f" {team_1} vs {team_2} \n  Results of the Match {res[0]}\n   {team_lst[win]} wins 🏆⚽🎖\n")
                if (win)==0:
                    team_wins_dct[team_1] = team_wins_dct.get(team_1,0)+2
                    team_wins_dct[team_2] = team_wins_dct.get(team_2,0)

                    win_dct[team_1] = win_dct.get(team_1,0)+1
                    win_dct[team_2] = win_dct.get(team_2,0)
                    lost_dct[team_2] = lost_dct.get(team_2,0)+1
                    lost_dct[team_1] = lost_dct.get(team_1,0)
                    draw_dct[team_2] = draw_dct.get(team_2,0)
                    draw_dct[team_1] = draw_dct.get(team_1,0)

                elif (win)==1:
                    team_wins_dct[team_2] = team_wins_dct.get(team_2,0)+2
                    team_wins_dct[team_1] = team_wins_dct.get(team_1,0)

                    win_dct[team_2] = win_dct.get(team_2,0)+1
                    win_dct[team_1] = win_dct.get(team_1,0)
                    lost_dct[team_1] = lost_dct.get(team_1,0)+1
                    lost_dct[team_2] = lost_dct.get(team_2,0)
                    draw_dct[team_1] = draw_dct.get(team_1,0)
                    draw_dct[team_2] = draw_dct.get(team_2,0)

            except IndexError:
                print(f"{team_1} vs {team_2} \n  Results of the Match {res[0]}\n   Match Draw ⚽⚽⚽\n")
                team_wins_dct[team_1] = team_wins_dct.get(team_1,0)+1
                team_wins_dct[team_2] = team_wins_dct.get(team_2,0)+1

                draw_dct[team_1] = draw_dct.get(team_1,0)+1
                draw_dct[team_2] = draw_dct.get(team_2,0)+1

                win_dct[team_1] = win_dct.get(team_1,0)
                lost_dct[team_1] = lost_dct.get(team_1,0)

                win_dct[team_2] = win_dct.get(team_2,0)
                lost_dct[team_2] = lost_dct.get(team_2,0)

            j=j+1
    group_results=[win_dct,draw_dct,lost_dct,team_wins_dct,goal_scored_dct,goal_against_dct]
    Group_standings[grp_name]=group_results
```

```
Group A Matches
 Uruguay vs Russia
  Results of the Match [1 0]
   Uruguay wins 🏆⚽🎖️

 Uruguay vs Saudi Arabia
  Results of the Match [2 0]
   Uruguay wins 🏆⚽🎖️

 Uruguay vs Egypt
  Results of the Match [1 0]
   Uruguay wins 🏆⚽🎖️

 Russia vs Saudi Arabia
  Results of the Match [1 0]
   Russia wins 🏆⚽🎖️


/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomFores
tClassifier was fitted with feature names
```

In [97]:
```
#Display group stage results

for grp_name in list(Group_standings.keys()):

    team_wins_dct= dict(sorted(Group_standings[grp_name][3].items()))
    goal_scored_dct=dict(sorted(Group_standings[grp_name][4].items()))
    goal_against_dct=dict(sorted(Group_standings[grp_name][5].items()))

    win_dct=dict(sorted(Group_standings[grp_name][0].items()))
    draw_dct=dict(sorted(Group_standings[grp_name][1].items()))
    lost_dct=dict(sorted(Group_standings[grp_name][2].items()))

    lst_teams=list(team_wins_dct.keys())

    win_lst=list(win_dct.values())
    draw_lst=list(draw_dct.values())
    lost_lst=list(lost_dct.values())

    lst_win_count=list(team_wins_dct.values())
    goal_scored=list(goal_scored_dct.values())
    goal_against=list(goal_against_dct.values())
    goal_differance=[goal_scored[i]-goal_against[i] for i in range (len(goal_scored))]
    ranking_table=pd.DataFrame(list(zip(lst_teams,win_lst,draw_lst,lost_lst,goal_scored,goal_against,goal_differance,lst_win_cour
    ranking_table=ranking_table.sort_values("Points",ascending=False).reset_index(drop=True)
    ranking_table.index = ranking_table.index + 1
    print(f"\n\n{grp_name} Final Rankings")
    print(ranking_table.to_markdown())
```

Group A Final Rankings

	Team	Wins	Draw	Lost	Goal Scored	Goal Against	Goal Differance	Points
1	Uruguay	3	0	0	4	0	4	6
2	Egypt	2	0	1	2	1	1	4
3	Russia	1	0	2	1	2	-1	2
4	Saudi Arabia	0	0	3	0	4	-4	0

Group B Final Rankings

	Team	Wins	Draw	Lost	Goal Scored	Goal Against	Goal Differance	Points
1	Portugal	1	2	0	2	1	1	4
2	Iran	0	3	0	2	2	0	3
3	Morocco	1	1	1	1	1	0	3
4	Spain	0	2	1	1	2	-1	2

Group C Final Rankings

	Team	Wins	Draw	Lost	Goal Scored	Goal Against	Goal Differance	Points
1	Australia	2	1	0	2	0	2	5
2	Denmark	1	1	1	2	2	0	3
3	France	1	0	2	1	2	-1	2
4	Peru	0	2	1	1	2	-1	2

Group D Final Rankings

	Team	Wins	Draw	Lost	Goal Scored	Goal Against	Goal Differance	Points
1	Argentina	2	1	0	3	1	2	5
2	Iceland	1	1	1	2	2	0	3
3	Nigeria	1	1	1	3	3	0	3
4	Croatia	0	1	2	0	2	-2	1

Group E Final Rankings

	Team	Wins	Draw	Lost	Goal Scored	Goal Against	Goal Differance	Points
1	Costa Rica	1	2	0	2	1	1	4
2	Brazil	1	1	1	1	1	0	3
3	Switzerland	1	1	1	3	3	0	3
4	Serbia	0	2	1	1	2	-1	2

Group F Final Rankings

	Team	Wins	Draw	Lost	Goal Scored	Goal Against	Goal Differance	Points
1	Sweden	2	1	0	3	1	2	5
2	Germany	0	3	0	3	3	0	3
3	South Korea	1	1	1	3	3	0	3
4	Mexico	0	1	2	2	4	-2	1

Group G Final Rankings

	Team	Wins	Draw	Lost	Goal Scored	Goal Against	Goal Differance	Points
1	England	2	1	0	3	1	2	5
2	Panama	1	2	0	2	1	1	4
3	Belgium	1	0	2	1	2	-1	2
4	Tunisia	0	1	2	0	2	-2	1

Group H Final Rankings

	Team	Wins	Draw	Lost	Goal Scored	Goal Against	Goal Differance	Points
1	Colombia	2	1	0	2	0	2	5
2	Japan	2	1	0	2	0	2	5
3	Poland	0	1	2	2	4	-2	1
4	Senegal	0	1	2	2	4	-2	1

Observation Above table is the result of group stage.

Q19. List all the team which qualifies through Round of 16.

In [98]:
```
##Round of 16 Section_1

qualified_teams_1=[]
standings=list(Group_standings.keys())
i=0
print(f"Round of 16\n")
while i < (len(standings)):
    A_team= sorted(Group_standings[standings[i]][3].items(), key=lambda x: x[1], reverse=True)
    team_1=A_team[0][0]
    B_team= sorted(Group_standings[standings[i+1]][3].items(), key=lambda x: x[1], reverse=True)
    team_2=B_team[1][0]

    team_1_num=label_encoder.transform([team_1])[0]
    team_2_num=label_encoder.transform([team_2])[0]
    team_lst=[team_1,team_2]

    Input_vector=np.array([[year,host_num,team_1_num,team_2_num]])
    res=model.predict(Input_vector)
    win,_=select_winning_team(res)

    try:
            print(f"{team_1} vs {team_2} \n  {team_lst[win]} wins 🏆🏆")
            print(f"    {team_lst[win]} into the Quater-Finals ⏩⏩ \n")
            qualified_teams_1.append(team_lst[win])
    except IndexError:
            print(f"{team_1} vs {team_2} \n  Match Draw ⚽⚽")
            winning_team=random.choice(team_lst)
            print(f"   {winning_team} wins at Penaly Shoot-Out 🥅🎯")
            print(f"    {winning_team} into the Quater-Finals ⏩⏩ \n")
            qualified_teams_1.append(winning_team)
    i=i+2

##Round of 16 Section_2
qualified_teams_2=[]
standings=list(Group_standings.keys())
i=0
while i < (len(standings)):
    A_team= sorted(Group_standings[standings[i]][3].items(), key=lambda x: x[1], reverse=True)
    team_1=A_team[1][0]
    B_team= sorted(Group_standings[standings[i+1]][3].items(), key=lambda x: x[1], reverse=True)
    team_2=B_team[0][0]

    team_1_num=label_encoder.transform([team_1])[0]
    team_2_num=label_encoder.transform([team_2])[0]
    team_lst=[team_1,team_2]

    Input_vector=np.array([[year,host_num,team_1_num,team_2_num]])
    res=model.predict(Input_vector)
    win,_=select_winning_team(res)

    try:
            print(f"{team_1} vs {team_2} \n  {team_lst[win]} wins 🏆🏆")
            print(f"    {team_lst[win]} into the Quater-Finals ⏩⏩ \n")
            qualified_teams_2.append(team_lst[win])

    except IndexError:
            print(f"{team_1} vs {team_2} \n  Match Draw ⚽⚽")
            winning_team=random.choice(team_lst)
            print(f"   {winning_team} wins at Penaly Shoot-Out 🥅🎯")
            print(f"    {winning_team} into the Quater-Finals ⏩⏩ \n")
            qualified_teams_2.append(winning_team)
    i=i+2
```

```
Round of 16

Uruguay vs Morocco
  Match Draw ⚽⚽
   Uruguay wins at Penaly Shoot-Out 🥅🎯
    Uruguay into the Quater-Finals ⏩⏩

Australia vs Iceland
  Match Draw ⚽⚽
   Iceland wins at Penaly Shoot-Out 🥅🎯
    Iceland into the Quater-Finals ⏩⏩

Costa Rica vs South Korea
  Costa Rica wins 🏆🏆
    Costa Rica into the Quater-Finals ⏩⏩

England vs Colombia
  Match Draw ⚽⚽
   Colombia wins at Penaly Shoot-Out 🥅🎯
    Colombia into the Quater-Finals ⏩⏩

Egypt vs Portugal
  Match Draw ⚽⚽
   Egypt wins at Penaly Shoot-Out 🥅🎯
    Egypt into the Quater-Finals ⏩⏩

Denmark vs Argentina
  Argentina wins 🏆🏆
    Argentina into the Quater-Finals ⏩⏩

Brazil vs Sweden
  Brazil wins 🏆🏆
    Brazil into the Quater-Finals ⏩⏩


/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
```

```
Panama vs Japan
  Japan wins 🏆🏆
    Japan into the Quater-Finals ⏩⏩
```

In [99]:
```
print(f"Teams selected to the Quater Finals - {qualified_teams_1+qualified_teams_2}")
```

```
Teams selected to the Quater Finals - ['Uruguay', 'Iceland', 'Costa Rica', 'Colombia', 'Egypt', 'Argentina', 'Brazil', 'Japan']
```

Observation : The teams which qualified through quater final.

Q20. List the team which are qualified in Quarter Finals.

In [100]:
```
#Quarter Finals

Semifinal_teams=[]
i=0
print(f"Quater Final Matches\n")
while i < (len(qualified_teams_1))-1:
    team_1= qualified_teams_1[i]
    team_2= qualified_teams_1[i+1]

    team_1_num=label_encoder.transform([team_1])[0]
    team_2_num=label_encoder.transform([team_2])[0]
    team_lst=[team_1,team_2]

    Input_vector=np.array([[year,host_num,team_1_num,team_2_num]])
    res=model.predict(Input_vector)
    win,_=select_winning_team(res)

    try:
            print(f"{team_1} vs {team_2} \n  {team_lst[win]} wins 🏆🏆")
            print(f"    {team_lst[win]} into the Semi-Finals ⏩⏩ \n")
            Semifinal_teams.append(team_lst[win])

    except IndexError:
            print(f"{team_1} vs {team_2} \n  Match Draw ⚽⚽")
            winning_team=random.choice(team_lst)
            print(f"   {winning_team} wins at Penaly Shoot-Out 🥅🎯")
            print(f"    {winning_team} into the Semi-Finals ⏩⏩ \n")
            Semifinal_teams.append(winning_team)
    i=i+2

i=0
while i < (len(qualified_teams_2))-1:
    team_1= qualified_teams_2[i]
    team_2= qualified_teams_2[i+1]
    team_1_num=label_encoder.transform([team_1])[0]
    team_2_num=label_encoder.transform([team_2])[0]
    team_lst=[team_1,team_2]

    Input_vector=np.array([[year,host_num,team_1_num,team_2_num]])
    res=model.predict(Input_vector)
    win,_=select_winning_team(res)

    try:
            print(f"{team_1} vs {team_2} \n  {team_lst[win]} wins 🏆🏆")
            print(f"    {team_lst[win]} into the Semi-Finals ⏩⏩ \n")
            Semifinal_teams.append(team_lst[win])

    except IndexError:
            print(f"{team_1} vs {team_2} \n  Match Draw ⚽⚽")
            winning_team=random.choice(team_lst)
            print(f"   {winning_team} wins at Penaly Shoot-Out 🥅🎯")
            print(f"    {winning_team} into the Semi-Finals ⏩⏩ \n")
            Semifinal_teams.append(winning_team)
    i=i+2
```

```
Quater Final Matches

Uruguay vs Iceland
  Match Draw ⚽⚽
   Uruguay wins at Penaly Shoot-Out 🥅🎯
    Uruguay into the Semi-Finals ⏩⏩

Costa Rica vs Colombia
  Colombia wins 🏆🏆
    Colombia into the Semi-Finals ⏩⏩

Egypt vs Argentina
  Argentina wins 🏆🏆
    Argentina into the Semi-Finals ⏩⏩

Brazil vs Japan
  Japan wins 🏆🏆
    Japan into the Semi-Finals ⏩⏩
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
```

In [101]:
```
print(f"Teams selected to the Semi-Finals - {Semifinal_teams}")
```

```
Teams selected to the Semi-Finals - ['Uruguay', 'Colombia', 'Argentina', 'Japan']
```

Observation : Teams which qualified for the semifinals.

Q21. List the team which are qualified in SemiFinals.

In [102]:
```
#Semi Finals

final_teams=[]
third_place_match_teams=[]
i=0
print(f"Semi Final Matches\n")
while i < (len(Semifinal_teams))-1:
    team_1= Semifinal_teams[i]
    team_2= Semifinal_teams[i+1]

    team_1_num=label_encoder.transform([team_1])[0]
    team_2_num=label_encoder.transform([team_2])[0]
    team_lst=[team_1,team_2]

    Input_vector=np.array([[year,host_num,team_1_num,team_2_num]])
    res=model.predict(Input_vector)
    win,_=select_winning_team(res)

    try:
            print(f"{team_1} vs {team_2} \n  {team_lst[win]} wins 🏆🏆")
            print(f"    {team_lst[win]} into the FiIFA-Finals ⏩⏩ \n")
            final_teams.append(team_lst[win])
            third_place_match_teams.append(team_lst[(win+1)%2])


    except IndexError:
            print(f"{team_1} vs {team_2} \n  Match Draw ⚽⚽")
            winning_team=random.choice(team_lst)
            print(f"   {winning_team} wins at Penaly Shoot-Out 🥅🎯")
            print(f"    {winning_team} into the FIFA-Finals ⏩⏩ \n")
            final_teams.append(winning_team)
            team_lst.remove(winning_team)
            third_place_match_teams.append(team_lst[0])
    i=i+2
```

```
Semi Final Matches

Uruguay vs Colombia
  Colombia wins 🏆🏆
    Colombia into the FiIFA-Finals ⏩⏩

Argentina vs Japan
  Match Draw ⚽⚽
   Japan wins at Penaly Shoot-Out 🥅🎯
    Japan into the FIFA-Finals ⏩⏩


/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
```

In [103]:
```
print(f"Teams selected to the Finals - {final_teams}")
print(f"Teams selected to the Third Place match - {third_place_match_teams}")
```

```
Teams selected to the Finals - ['Colombia', 'Japan']
Teams selected to the Third Place match - ['Uruguay', 'Argentina']
```

Observation : Teams selected for Finals, and Third place match.

Q22. Who is the winner? Q23. Who is qualified of Third Place?

In [104]:
```
#Finals and Third Place match

print(f"FIFA FINAl\n")
team_1= final_teams[1]
team_2= final_teams[0]

team_1_num=label_encoder.transform([team_1])[0]
team_2_num=label_encoder.transform([team_2])[0]
team_lst=[team_1,team_2]

Input_vector=np.array([[year,host_num,team_1_num,team_2_num]])
res=model.predict(Input_vector)
win,_=select_winning_team(res)

try:
    print(f"{team_1} vs {team_2} \n  {team_lst[win]} are the Winners \n\n")
    winner=team_lst[win]
    place_2=team_lst[(win+1)%2]

except IndexError:
    print(f"{team_1} vs {team_2} \n  Match Draw ")
    winning_team=random.choice(team_lst)
    print(f"   {winning_team} wins at Penaly Shoot-Out ")
    print(f"    {winning_team} are the Winners \n\n")
    winner=winning_team

    team_lst.remove(winning_team)
    place_2=team_lst[0]

print(f"Third Place match\n")
team_1= third_place_match_teams[1]
team_2= third_place_match_teams[0]

team_1_num=label_encoder.transform([team_1])[0]
team_2_num=label_encoder.transform([team_2])[0]
team_lst=[team_1,team_2]

Input_vector=np.array([[year,host_num,team_1_num,team_2_num]])
res=model.predict(Input_vector)
win,_=select_winning_team(res)

try:
    print(f"{team_1} vs {team_2} \n  {team_lst[win]} Wins the 3rd Place \n")
    place_3=team_lst[win]

except IndexError:
    print(f"{team_1} vs {team_2} \n  Match Draw ")
    winning_team=random.choice(team_lst)
    print(f"   {winning_team} wins at Penaly Shoot-Out ")
    print(f"    {winning_team} Wins the 3rd Place \n")
    place_3=winning_team



print(f"\n\nWinner is {winner} ")
print(f"Runner-up is {place_2} ")
print(f"3rd Place is {place_3} ")
```

```
FIFA FINAl

Japan vs Colombia
  Match Draw 
   Colombia wins at Penaly Shoot-Out 
    Colombia are the Winners 


Third Place match

Argentina vs Uruguay
  Argentina Wins the 3rd Place 


Winner is Colombia 
Runner-up is Japan 
3rd Place is Argentina 
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but RandomForestC
lassifier was fitted with feature names
  warnings.warn(
```

In [105]:
```
Image(filename='Fi.jpg')
```

Out[105]:

In [105]: