Submitted By:

> Richard Honey

FIFA 19 Analysis

In [49]:
```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
```

In [50]:
```
df =  pd.read_csv('data.csv',delimiter = ',')
```

Data Structure

In [51]:
```
df.head()
```

Out[51]:

	Unnamed: 0	ID	Name	Age	Photo	Nationality	
0	0	158023	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	Argentina	https://cdn.sofifa.org/fla
1	1	20801	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	Portugal	https://cdn.sofifa.org/fla
2	2	190871	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	Brazil	https://cdn.sofifa.org/fla
3	3	193080	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	Spain	https://cdn.sofifa.org/fla
4	4	192985	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	Belgium	https://cdn.sofifa.org/fl

5 rows × 89 columns

In [52]:
```
df.info()
```

Loading [MathJax]/extensions/Safe.js

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18207 entries, 0 to 18206
Data columns (total 89 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Unnamed: 0                18207 non-null  int64  
 1   ID                        18207 non-null  int64  
 2   Name                      18207 non-null  object 
 3   Age                       18207 non-null  int64  
 4   Photo                     18207 non-null  object 
 5   Nationality               18207 non-null  object 
 6   Flag                      18207 non-null  object 
 7   Overall                   18207 non-null  int64  
 8   Potential                 18207 non-null  int64  
 9   Club                      17966 non-null  object 
 10  Club Logo                 18207 non-null  object 
 11  Value                     18207 non-null  object 
 12  Wage                      18207 non-null  object 
 13  Special                   18207 non-null  int64  
 14  Preferred Foot            18159 non-null  object 
 15  International Reputation  18159 non-null  float64
 16  Weak Foot                 18159 non-null  float64
 17  Skill Moves               18159 non-null  float64
 18  Work Rate                 18159 non-null  object 
 19  Body Type                 18159 non-null  object 
 20  Real Face                 18159 non-null  object 
 21  Position                  18147 non-null  object 
 22  Jersey Number             18147 non-null  float64
 23  Joined                    16654 non-null  object 
 24  Loaned From               1264 non-null   object 
 25  Contract Valid Until      17918 non-null  object 
 26  Height                    18159 non-null  object 
 27  Weight                    18159 non-null  object 
 28  LS                        16122 non-null  object 
 29  ST                        16122 non-null  object 
 30  RS                        16122 non-null  object 
 31  LW                        16122 non-null  object 
 32  LF                        16122 non-null  object 
 33  CF                        16122 non-null  object 
 34  RF                        16122 non-null  object 
 35  RW                        16122 non-null  object 
 36  LAM                       16122 non-null  object 
 37  CAM                       16122 non-null  object 
 38  RAM                       16122 non-null  object 
 39  LM                        16122 non-null  object 
 40  LCM                       16122 non-null  object 
 41  CM                        16122 non-null  object 
 42  RCM                       16122 non-null  object 
 43  RM                        16122 non-null  object 
 44  LWB                       16122 non-null  object 
 45  LDM                       16122 non-null  object 
 46  CDM                       16122 non-null  object 
 47  RDM                       16122 non-null  object 
 48  RWB                       16122 non-null  object 
 49  LB                        16122 non-null  object 
 50  LCB                       16122 non-null  object 
 51  CB                        16122 non-null  object 
 52  RCB                       16122 non-null  object 
 53  RB                        16122 non-null  object 
 54  Crossing                  18159 non-null  float64
 55  Finishing                 18159 non-null  float64
 56  HeadingAccuracy           18159 non-null  float64
 57  ShortPassing              18159 non-null  float64
 58  Volleys                   18159 non-null  float64
```

Loading [MathJax]/extensions/Safe.js

```
 59  Dribbling                 18159 non-null  float64
 60  Curve                     18159 non-null  float64
 61  FKAccuracy                18159 non-null  float64
 62  LongPassing               18159 non-null  float64
 63  BallControl               18159 non-null  float64
 64  Acceleration              18159 non-null  float64
 65  SprintSpeed               18159 non-null  float64
 66  Agility                   18159 non-null  float64
 67  Reactions                 18159 non-null  float64
 68  Balance                   18159 non-null  float64
 69  ShotPower                 18159 non-null  float64
 70  Jumping                   18159 non-null  float64
 71  Stamina                   18159 non-null  float64
 72  Strength                  18159 non-null  float64
 73  LongShots                 18159 non-null  float64
 74  Aggression                18159 non-null  float64
 75  Interceptions             18159 non-null  float64
 76  Positioning               18159 non-null  float64
 77  Vision                    18159 non-null  float64
 78  Penalties                 18159 non-null  float64
 79  Composure                 18159 non-null  float64
 80  Marking                   18159 non-null  float64
 81  StandingTackle            18159 non-null  float64
 82  SlidingTackle             18159 non-null  float64
 83  GKDiving                  18159 non-null  float64
 84  GKHandling                18159 non-null  float64
 85  GKKicking                 18159 non-null  float64
 86  GKPositioning             18159 non-null  float64
 87  GKReflexes                18159 non-null  float64
 88  Release Clause            16643 non-null  object
dtypes: float64(38), int64(6), object(45)
memory usage: 12.4+ MB
```

In [53]:
```
df.shape
```

Out[53]:
```
(18207, 89)
```

Data Pre-processing

Deleting Columns

In [54]:
```
df = df.drop(columns="Unnamed: 0")
```

In [55]:
```
columns = ['Photo', 'Flag', 'Club Logo', 'Release Clause', 'Jersey Number', 'Loaned From
       'CM', 'RCM', 'RM', 'LWB', 'LDM', 'CDM', 'RDM', 'RWB', 'LB', 'LCB', 'CB',
       'RCB', 'RB', 'LS', 'ST']
df = df.drop(columns, axis=1, inplace=False )
```

In [56]:
```
column = ['Crossing', 'Finishing', 'HeadingAccuracy', 'ShortPassing', 'Volleys',
       'Dribbling', 'Curve', 'FKAccuracy', 'LongPassing', 'BallControl',
       'Acceleration', 'SprintSpeed', 'Agility', 'Reactions', 'Balance',
       'ShotPower', 'Jumping', 'Stamina', 'Strength', 'LongShots',
       'Aggression', 'Interceptions', 'Positioning', 'Vision', 'Penalties',
       'Composure', 'Marking', 'StandingTackle', 'SlidingTackle', 'GKDiving',
       'GKHandling', 'GKKicking', 'GKPositioning', 'GKReflexes']

df = df.drop(column, axis=1, inplace=False )
```

Checking the columns

Loading [MathJax]/extensions/Safe.js

In [57]:
```
df.columns
```
Out[57]:
```
Index(['ID', 'Name', 'Age', 'Nationality', 'Overall', 'Potential', 'Club',
       'Value', 'Wage', 'Special', 'Preferred Foot',
       'International Reputation', 'Weak Foot', 'Skill Moves', 'Work Rate',
       'Body Type', 'Real Face', 'Position', 'Joined', 'Contract Valid Until',
       'Height', 'Weight'],
      dtype='object')
```

Data Conversion

Height Conversion from inch to centimeter

In [58]:
```
#in centimeter
def height_conversion(height):
    if(pd.isna(height))!= True:
        chk = str(height)
        h = []
        h = chk.split("'")
        ft = float(h[0])
        if( h[1] != ''):
            inch = float(h[1])
        else:
            inch = 0
        tot_inc = inch + ft*12
        h = tot_inc * 2.54
        return h
    else:
        return height

df['Height'] = df['Height'].apply(height_conversion)
```

Weight conversion: lbs to kg

In [59]:
```
#in kg
def weight_conversion(weight):
    if(pd.isna(weight))!= True:
        w = int(weight[0:-3])*0.453592
        return w
    else:
        return weight

df['Weight'] = df['Weight'].apply(weight_conversion)
```

Getting rid of all the elements that makes difficult to convert the different columns datatypes

In [60]:
```
df['Value'] = df['Value'].str.replace('€', '')
df['Value'] = df['Value'].str.replace('M', '')
df['Value'] = df['Value'].str.replace('K', '000')
df['Wage'] = df['Wage'].str.replace('€', '')
df['Wage'] = df['Wage'].str.replace('K', '000')
```

Renaming Columns

In [61]:
```
df.rename(columns = {'Value':"Value(millions)"}, inplace = True)
```

Changing the datatypes of the selected columns

Loading [MathJax]/extensions/Safe.js

In [62]:
```
df = df.astype({"Name":'category', "Value(millions)":'float', "Wage":'int64'})
```

Changing the datatype of date

In [63]:
```
df['Joined'] = pd.to_datetime(df['Joined'])
```

Treating Null Values

Checking for Null values

In [64]:
```
df.columns[df.isnull().any()]
```

Out[64]:
```
Index(['Club', 'Preferred Foot', 'International Reputation', 'Weak Foot',
       'Skill Moves', 'Work Rate', 'Body Type', 'Real Face', 'Position',
       'Joined', 'Contract Valid Until', 'Height', 'Weight'],
      dtype='object')
```

In [65]:
```
df.isnull().sum()
```

Out[65]:
```
ID                             0
Name                           0
Age                            0
Nationality                    0
Overall                        0
Potential                      0
Club                         241
Value(millions)                0
Wage                           0
Special                        0
Preferred Foot                48
International Reputation      48
Weak Foot                     48
Skill Moves                   48
Work Rate                     48
Body Type                     48
Real Face                     48
Position                      60
Joined                      1553
Contract Valid Until         289
Height                        48
Weight                        48
dtype: int64
```

Replacing Null values with most frequent values

In [66]:
```
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy='most_frequent', missing_values=np.nan)
imputer = imputer.fit(df[['Club']])
df['Club'] = imputer.transform(df[['Club']])

imputer = SimpleImputer(strategy='most_frequent', missing_values=np.nan)
imputer = imputer.fit(df[['Preferred Foot']])
df['Preferred Foot'] = imputer.transform(df[['Preferred Foot']])

imputer = SimpleImputer(strategy='most_frequent', missing_values=np.nan)
imputer = imputer.fit(df[['International Reputation']])
df['International Reputation'] = imputer.transform(df[['International Reputation']])

imputer = SimpleImputer(strategy='most_frequent', missing_values=np.nan)
```

Loading [MathJax]/extensions/Safe.js

```
imputer = imputer.fit(df[['Weak Foot']])
df['Weak Foot'] = imputer.transform(df[['Weak Foot']])

imputer = SimpleImputer(strategy='most_frequent', missing_values=np.nan)
imputer = imputer.fit(df[['Skill Moves']])
df['Skill Moves'] = imputer.transform(df[['Skill Moves']])

imputer = SimpleImputer(strategy='most_frequent', missing_values=np.nan)
imputer = imputer.fit(df[['Work Rate']])
df['Work Rate'] = imputer.transform(df[['Work Rate']])

imputer = SimpleImputer(strategy='most_frequent', missing_values=np.nan)
imputer = imputer.fit(df[['Body Type']])
df['Body Type'] = imputer.transform(df[['Body Type']])

imputer = SimpleImputer(strategy='most_frequent', missing_values=np.nan)
imputer = imputer.fit(df[['Real Face']])
df['Real Face'] = imputer.transform(df[['Real Face']])

imputer = SimpleImputer(strategy='most_frequent', missing_values=np.nan)
imputer = imputer.fit(df[['Position']])
df['Position'] = imputer.transform(df[['Position']])

imputer = SimpleImputer(strategy='most_frequent', missing_values=np.nan)
imputer = imputer.fit(df[['Contract Valid Until']])
df['Contract Valid Until'] = imputer.transform(df[['Contract Valid Until']])
```

Replacing null values by forward filling

In [67]:
```
df['Joined'] = df['Joined'].fillna(value = df['Joined'].ffill())
```

Replacing Null values with mean

In [68]:
```
df['Height'] = df['Height'].fillna(value = df['Height'].mean())
df['Weight'] = df['Weight'].fillna(value = df['Weight'].mean())
```

Checking for any Null values

In [69]:
```
df.columns[df.isnull().any()]
```

Out[69]:
```
Index([], dtype='object')
```

In [70]:
```
df.info()
```

Loading [MathJax]/extensions/Safe.js

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18207 entries, 0 to 18206
Data columns (total 22 columns):
 #   Column                    Non-Null Count  Dtype         
---  ------                    --------------  -----         
 0   ID                        18207 non-null  int64         
 1   Name                      18207 non-null  category      
 2   Age                       18207 non-null  int64         
 3   Nationality               18207 non-null  object        
 4   Overall                   18207 non-null  int64         
 5   Potential                 18207 non-null  int64         
 6   Club                      18207 non-null  object        
 7   Value(millions)           18207 non-null  float64       
 8   Wage                      18207 non-null  int64         
 9   Special                   18207 non-null  int64         
 10  Preferred Foot            18207 non-null  object        
 11  International Reputation  18207 non-null  float64       
 12  Weak Foot                 18207 non-null  float64       
 13  Skill Moves               18207 non-null  float64       
 14  Work Rate                 18207 non-null  object        
 15  Body Type                 18207 non-null  object        
 16  Real Face                 18207 non-null  object        
 17  Position                  18207 non-null  object        
 18  Joined                    18207 non-null  datetime64[ns]
 19  Contract Valid Until      18207 non-null  object        
 20  Height                    18207 non-null  float64       
 21  Weight                    18207 non-null  float64       
dtypes: category(1), datetime64[ns](1), float64(6), int64(6), object(8)
memory usage: 3.6+ MB
```

Saving the pre-processed data into an Excel sheet

In [71]:
```
df.to_csv('Pre-processed.csv')
```

EDA

Univariate Analysis

In [72]:
```
df.describe()
```

Out[72]:

	ID	Age	Overall	Potential	Value(millions)	Wage	Special
count	18207.000000	18207.000000	18207.000000	18207.000000	18207.000000	18207.000000	18207.000000
mean	214298.338606	25.122206	66.238699	71.307299	262881.260246	9731.312133	1597.809908
std	29965.244204	4.669943	6.908930	6.136496	291450.965245	21999.290406	272.586016
min	16.000000	16.000000	46.000000	48.000000	0.000000	0.000000	731.000000
25%	200315.500000	21.000000	62.000000	67.000000	4.400000	1000.000000	1457.000000
50%	221759.000000	25.000000	66.000000	71.000000	160000.000000	3000.000000	1635.000000
75%	236529.500000	28.000000	71.000000	75.000000	475000.000000	9000.000000	1787.000000
max	246620.000000	45.000000	94.000000	95.000000	975000.000000	565000.000000	2346.000000

Table of Indian footballers

Loading [MathJax]/extensions/Safe.js

In [73]:
```
def country(x):
    return df[df['Nationality'] == x][['Name','Overall','Potential','Position', 'Value(m

country('India')
```

Out[73]:

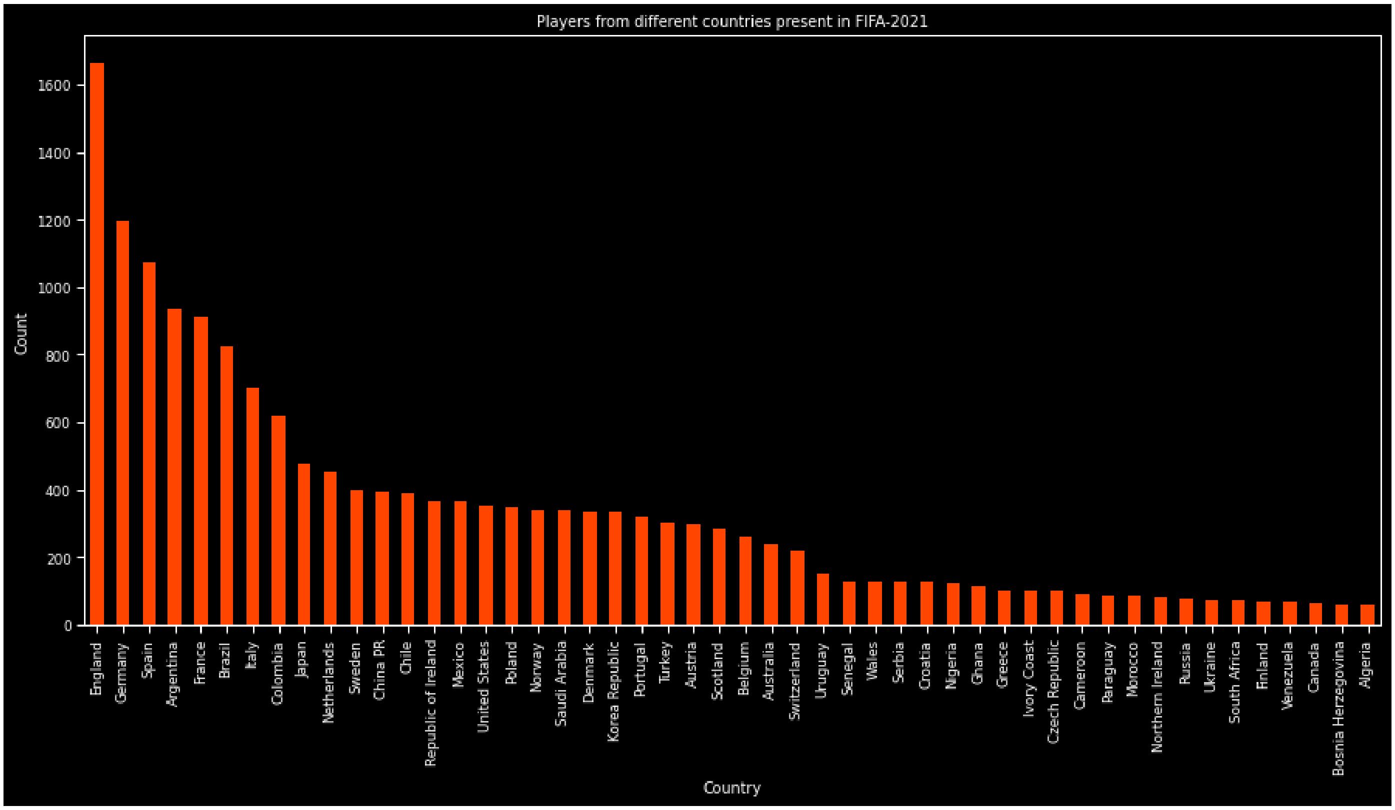
	Name	Overall	Potential	Position	Value(millions)	Height	Weight
8605	S. Chhetri	67	67	LS	0.0	170.18	69.853168
10011	S. Jhingan	65	71	RCB	0.0	187.96	73.935496
12598	J. Lalpekhlua	63	64	RS	0.0	175.26	74.842680
12811	G. Singh Sandhu	63	68	GK	0.0	193.04	89.811216
13508	A. Edathodika	62	62	LCB	0.0	182.88	78.017824
14054	P. Halder	61	67	RCM	0.0	180.34	73.935496
14199	P. Kotal	61	66	RB	0.0	177.80	73.935496
14218	L. Ralte	61	62	LW	0.0	172.72	71.213944
14705	N. Das	60	65	LB	0.0	175.26	68.038800
14786	U. Singh	60	67	RM	0.0	180.34	74.842680
14915	H. Narzary	60	66	LM	0.0	177.80	73.935496
15356	R. Singh	59	59	ST	0.0	185.42	74.842680
15643	S. Singh	59	65	CB	0.0	187.96	76.203456
15652	A. Thapa	59	71	LCM	0.0	170.18	63.956472
15855	M. Rafique	58	61	CM	0.0	172.72	67.131616
15864	A. Singh	58	62	GK	0.0	185.42	81.192968
15884	B. Singh	58	58	ST	0.0	180.34	71.213944
16135	S. Bose	58	66	LB	0.0	185.42	78.017824
16265	R. Borges	58	60	CDM	0.0	185.42	74.842680
16450	S. Paul	57	57	ST	0.0	185.42	78.017824
16499	A. Mondal	57	57	CB	0.0	177.80	69.853168
16539	L. Lalruatthara	57	63	ST	0.0	180.34	64.863656
16793	E. Lyngdoh	56	56	ST	0.0	175.26	68.038800
16903	J. Lalrinzuala	56	64	LB	0.0	175.26	68.038800
16976	A. Kuruniyan	56	70	LW	0.0	175.26	69.853168
17129	J. Singh	55	58	ST	0.0	170.18	72.121128
17197	V. Kaith	55	64	GK	0.0	187.96	79.832192
17339	S. Passi	54	63	ST	0.0	175.26	64.863656
17436	D. Lalhlimpuia	54	67	ST	0.0	182.88	76.203456
17539	C. Singh	53	62	ST	0.0	190.50	78.925008

Players from different countries present in FIFA-2021

In [74]:
```
plt.style.use('dark_background') #top 50 nations that the players represent in FIFA 2021
plt.figure(figsize = (15,7))
df['Nationality'].value_counts().head(50).plot.bar(color = 'orangered')
plt.title('Players from different countries present in FIFA-2021')
plt.xlabel('Country')
```

Loading [MathJax]/extensions/Safe.js

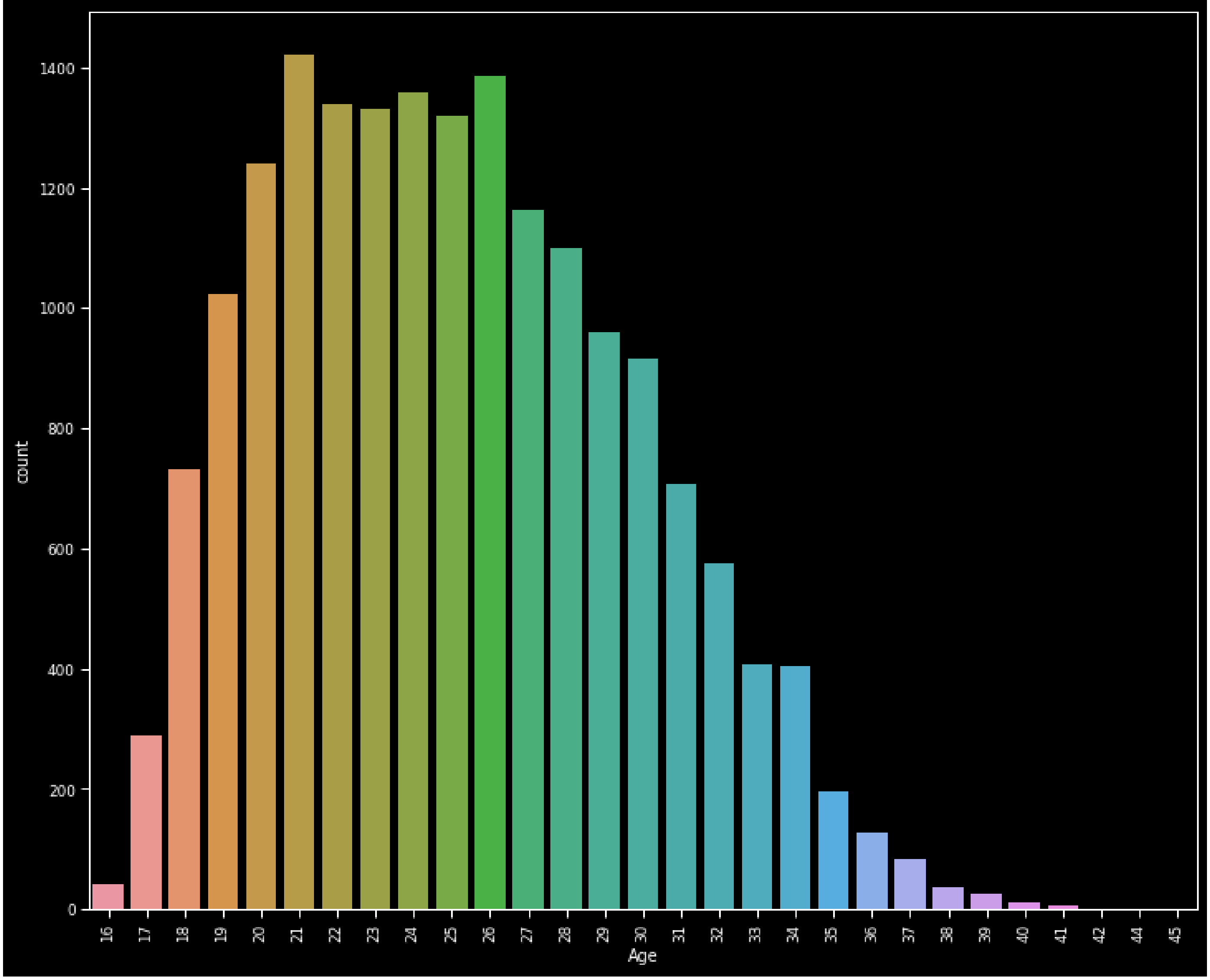
```
plt.ylabel('Count')
plt.show()
```



Ages in which maximum players are present

In [75]:
```
plt.figure(figsize=(12, 10))
sns.countplot(x=df.Age)
plt.xticks(rotation=90);
```

Loading [MathJax]/extensions/Safe.js

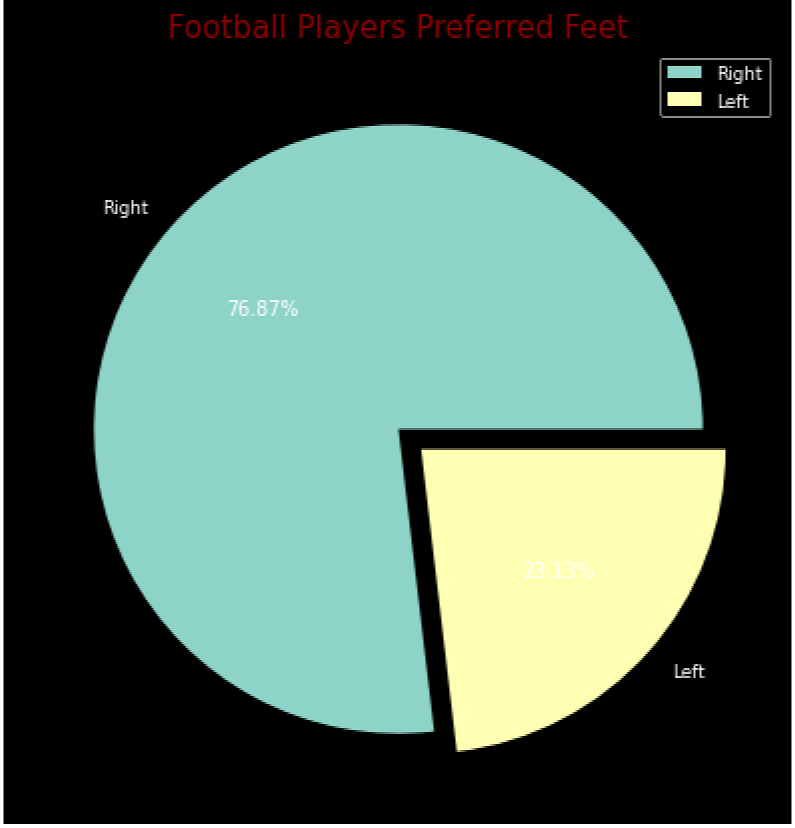


Word Cloud of nationalities of players in the shape of World Cup Trophy

Football players preferred feet

In [79]:
```
preferred_foot_labels = df["Preferred Foot"].value_counts().index # (Right,Left)
preferred_foot_values = df["Preferred Foot"].value_counts().values # (Right Values, Left
explode = (0, 0.1) # used to separate a slice of cake

# Visualize
plt.figure(figsize = (7,7))
plt.pie(preferred_foot_values, labels=preferred_foot_labels,explode=explode, autopct='%1
plt.title('Football Players Preferred Feet',color = 'darkred',fontsize = 15)
plt.legend()
plt.show()
```

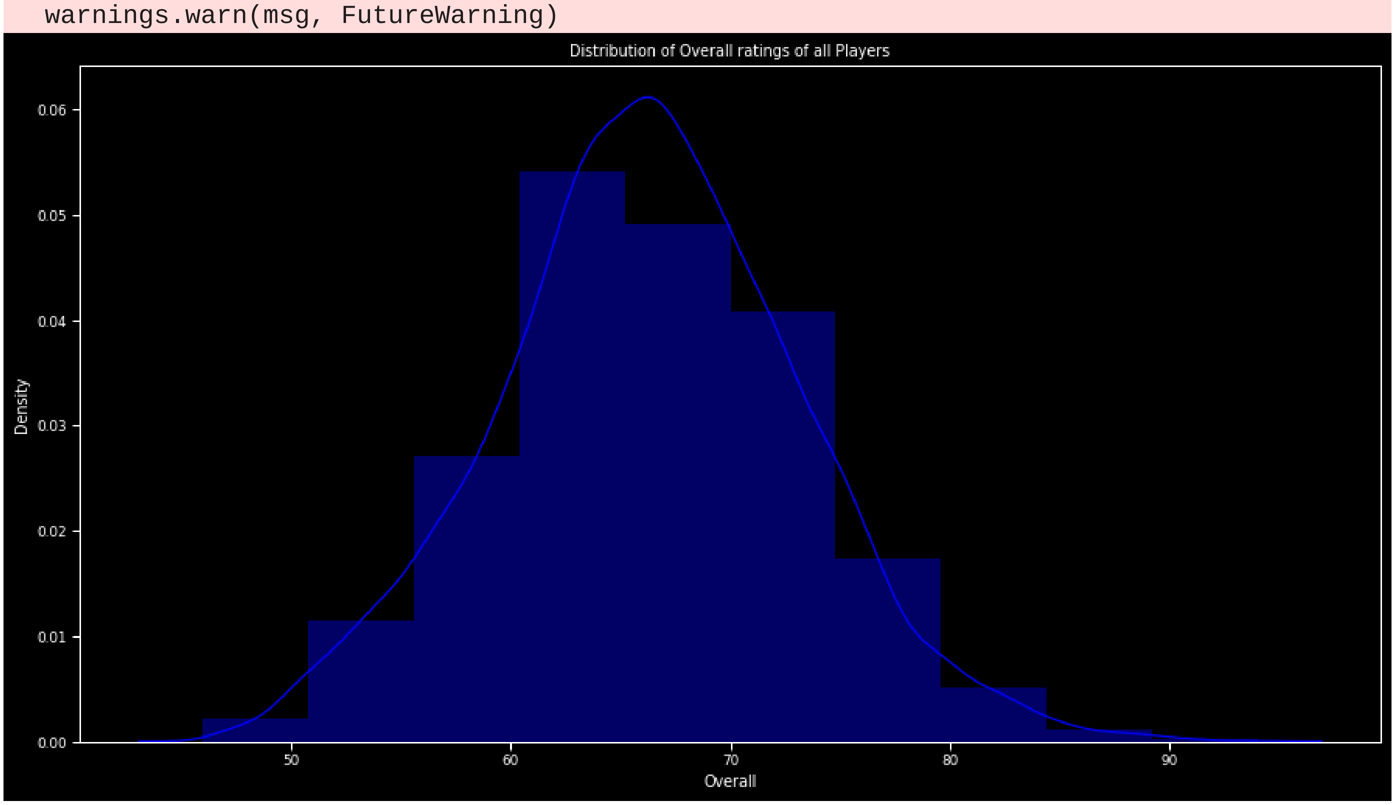
Loading [MathJax]/extensions/Safe.js



Distribution of overall rating for all players.

In [80]:
```
sns.distplot(df['Overall'], bins=10, color='b')
plt.title("Distribution of Overall ratings of all Players")
plt.show()
```

```
C:\Users\richa\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



Loading [MathJax]/extensions/Safe.js

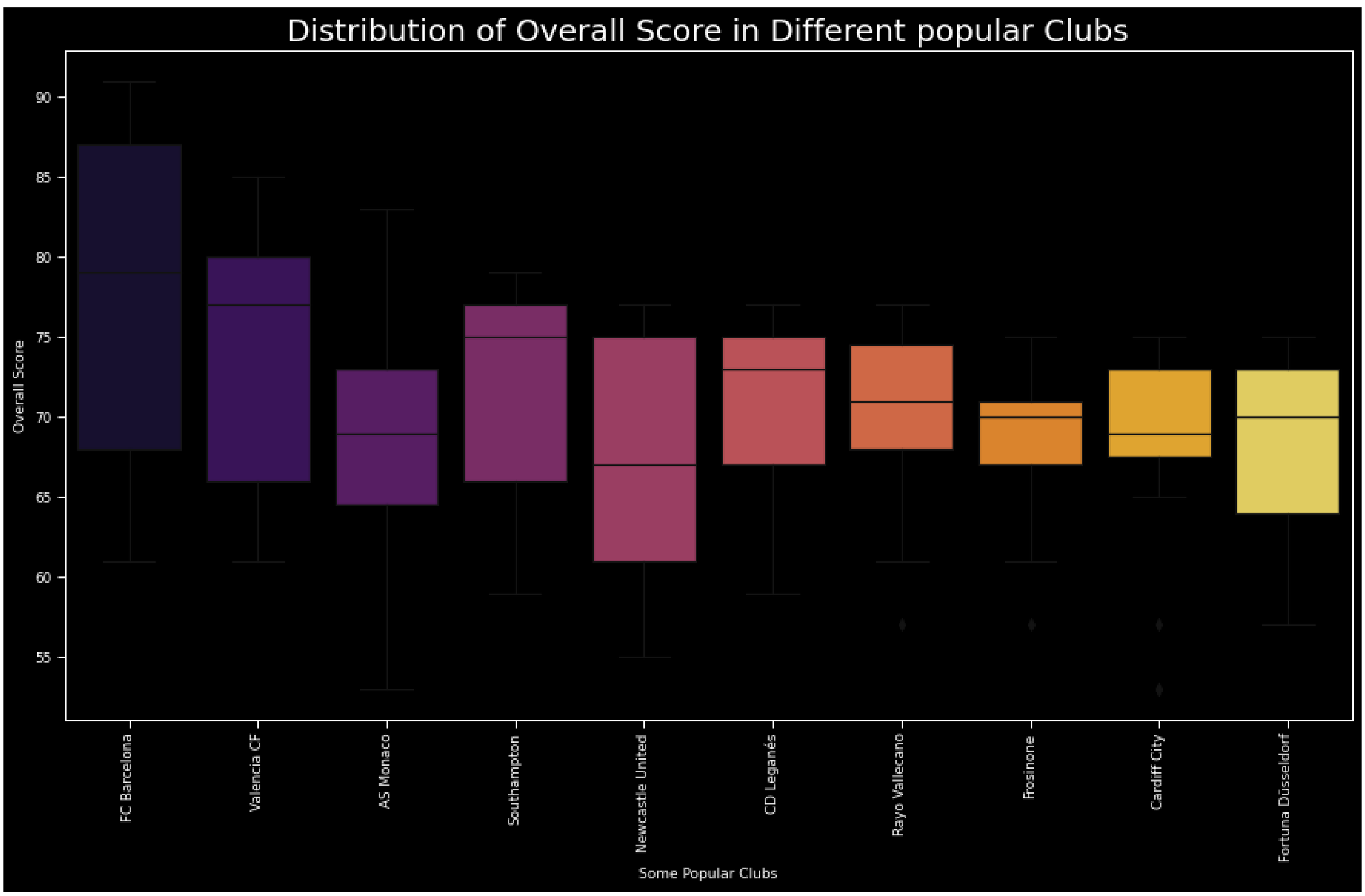
Popular clubs around the world

In [81]:
```
df['Club'].value_counts().head(10)
```

Out[81]:
```
AS Monaco              274
FC Barcelona            33
Valencia CF             33
Fortuna Düsseldorf      33
Cardiff City            33
Rayo Vallecano          33
CD Leganés              33
Frosinone               33
Newcastle United        33
Southampton             33
Name: Club, dtype: int64
```

Distribution of overall score in different popular clubs

In [82]:
```
some_clubs = ('AS Monaco', 'FC Barcelona', 'Valencia CF', 'Fortuna Düsseldorf', 'Cardiff
              'CD Leganés', 'Frosinone', 'Newcastle United', 'Southampton')

data_clubs = df.loc[df['Club'].isin(some_clubs) & df['Overall']]

plt.rcParams['figure.figsize'] = (15, 8)
ax = sns.boxplot(x = data_clubs['Club'], y = data_clubs['Overall'], palette = 'inferno')
ax.set_xlabel(xlabel = 'Some Popular Clubs', fontsize = 9)
ax.set_ylabel(ylabel = 'Overall Score', fontsize = 9)
ax.set_title(label = 'Distribution of Overall Score in Different popular Clubs', fontsiz
plt.xticks(rotation = 90)
plt.show()
```



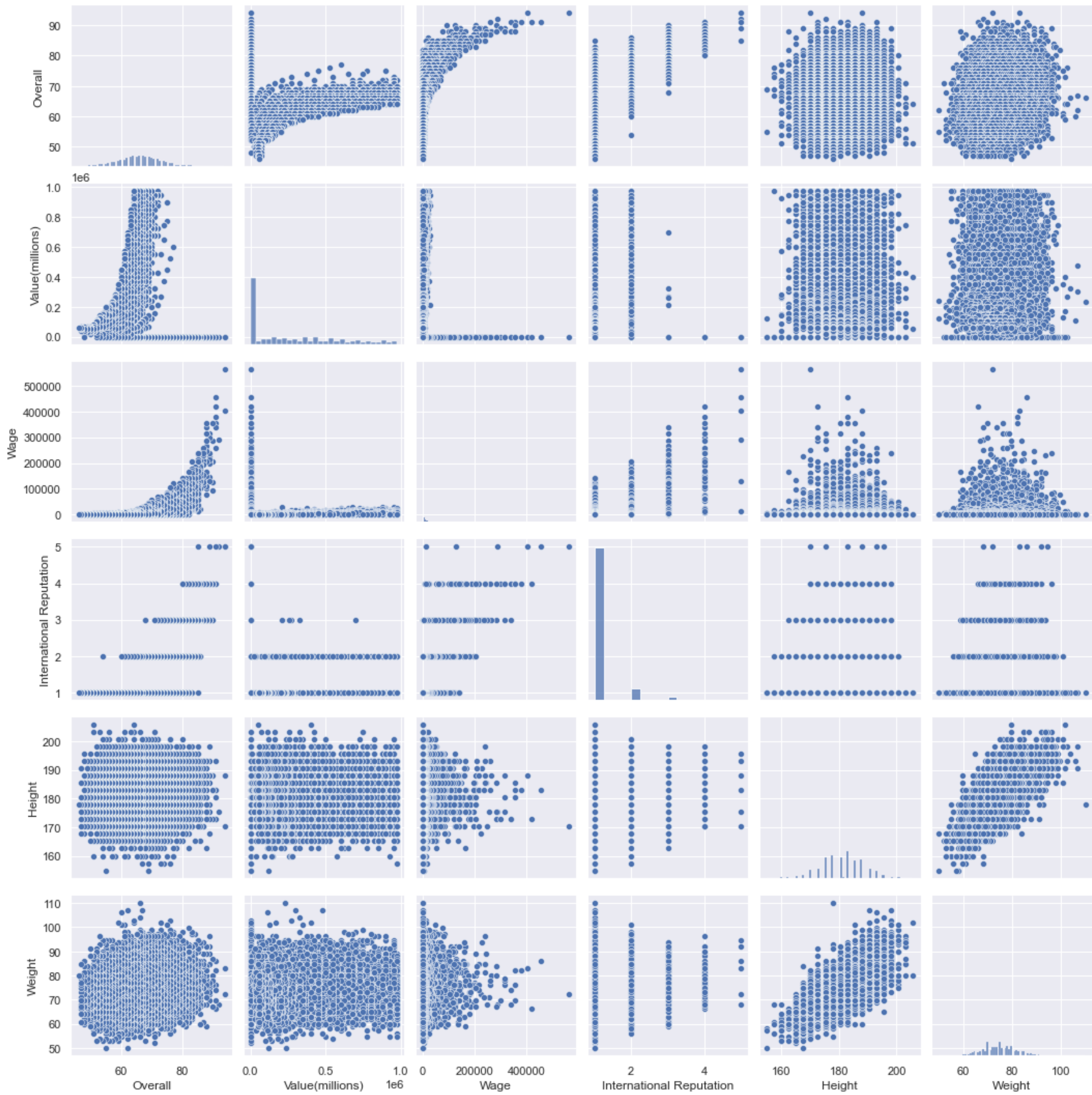
Bivariate Analysis

Loading [MathJax]/extensions/Safe.js

Pair plots for the following variables: Overall, Value(millions, Wage, International Reputation, Height and Weight

In [83]:

```
sns.set()
cols = ['Overall', 'Value(millions)', 'Wage', 'International Reputation', 'Height', 'Wei
sns.pairplot(df[cols], height = 2.5)
plt.show()
```

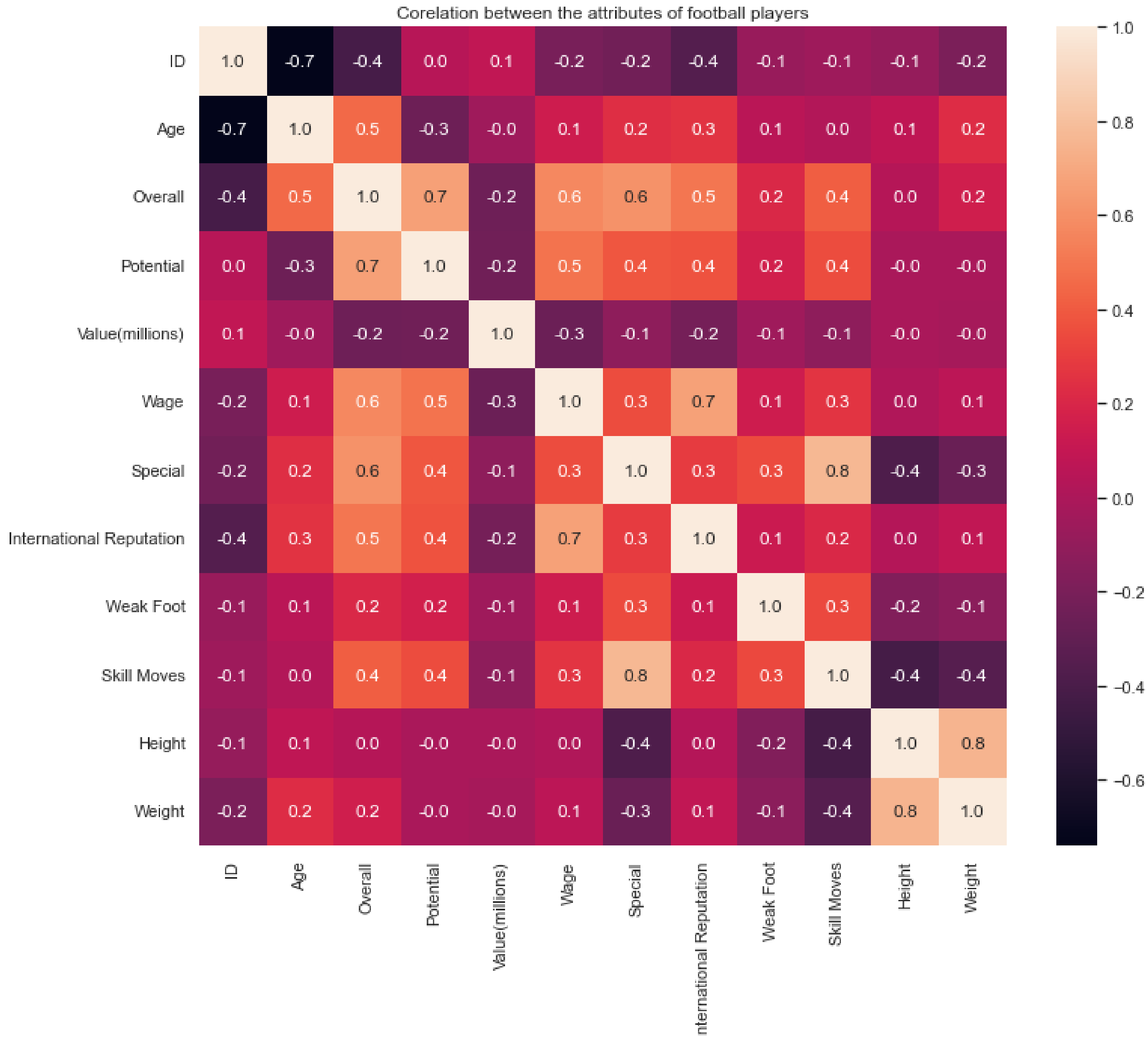


Heatmap of attributes of football players

In [84]:

```
import seaborn as sns
plt.figure(figsize = (12,10))
sns.heatmap(df.corr(), annot = True, fmt = '.1f')
plt.title("Corelation between the attributes of football players")
plt.show()
```

Loading [MathJax]/extensions/Safe.js



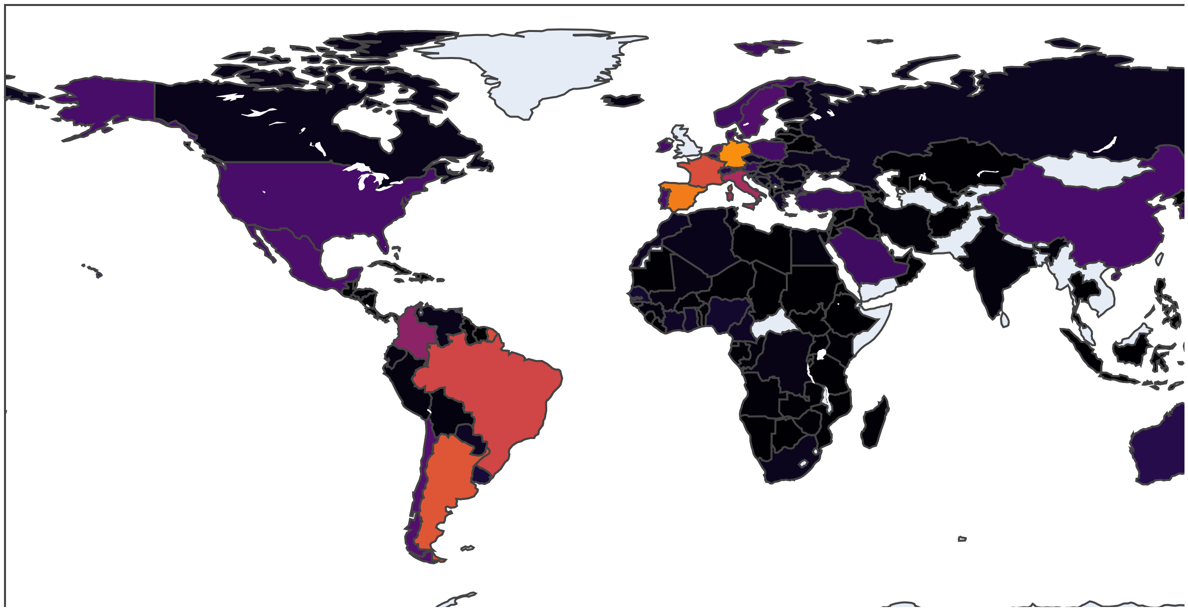
Country vs Overall Ratings of players belonging to them

In [85]:

```
import plotly.offline as py
from plotly.offline import init_notebook_mode, iplot
import plotly.graph_objs as go
rating = pd.DataFrame(df.groupby(['Nationality'])['Overall'].sum().reset_index())
count = pd.DataFrame(rating.groupby('Nationality')['Overall'].sum().reset_index())

plot = [go.Choropleth(
            colorscale = 'inferno',
            locationmode = 'country names',
            locations = count['Nationality'],
            text = count['Nationality'],
            z = count['Overall'],
)]

layout = go.Layout(title = 'Country vs Overall Ratings of players belonging to them')

fig = go.Figure(data = plot, layout = layout)
py.iplot(fig)
```

Loading [MathJax]/extensions/Safe.js

Country vs Overall Ratings of players belonging to them



Multivariate Analysis

Data Pre-processing for PCA and K-Mean Clustering

In [86]:
```
df2 = df.drop(columns= ['ID', 'Name', 'Nationality', 'Club', 'Value(millions)', 'Wage',
                 'Real Face','Position','Contract Valid Until', 'Height', 'Weight', 'Joined
df2.head()
```

Out[86]:

	Age	Overall	Potential	Special	International Reputation	Weak Foot	Skill Moves
0	31	94	94	2202	5.0	4.0	4.0
1	33	94	94	2228	5.0	4.0	5.0
2	26	92	93	2143	5.0	5.0	5.0
3	27	91	93	1471	4.0	3.0	1.0
4	27	91	92	2281	4.0	5.0	4.0

In [87]:
```
df2.dtypes
```

Loading [MathJax]/extensions/Safe.js

Out[87]:
```
Age                          int64
Overall                      int64
Potential                    int64
Special                      int64
International Reputation    float64
Weak Foot                   float64
Skill Moves                 float64
dtype: object
```

In [88]:
```
X = df2.values
# Using the standard scaler method to standardize all of the features by converting them
from sklearn.preprocessing import StandardScaler
X = StandardScaler().fit_transform(X)
X
```

Out[88]:
```
array([[ 1.25867833,  4.01828714,  3.69809177, ...,  9.87713252,
         1.59582491,  2.17064139],
       [ 1.68696087,  4.01828714,  3.69809177, ...,  9.87713252,
         1.59582491,  3.49449051],
       [ 0.18797198,  3.72879875,  3.53512784, ...,  9.87713252,
         3.1119585 ,  3.49449051],
       ...,
       [-1.95344072, -2.78469008, -0.70193445, ..., -0.28694094,
         0.07969132, -0.47705685],
       [-1.73929945, -2.78469008, -0.86489839, ..., -0.28694094,
         0.07969132, -0.47705685],
       [-1.95344072, -2.92943428, -0.86489839, ..., -0.28694094,
         0.07969132, -0.47705685]])
```

EDA for PCA and K-Mean Clustering

Principal Component Analysis

Using PCA to reduce dimensionality of the data

In [89]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=3)
principalComponents2 = pca.fit_transform(X)
```

Reduced features

In [90]:
```
principalComponents2
```

Out[90]:
```
array([[ 9.28997893,  1.9235749 ,  4.5423512 ],
       [ 9.96529346,  1.9230709 ,  3.95659477],
       [ 9.74199207,  0.49727381,  3.39826517],
       ...,
       [-2.96733896, -1.75093485, -0.23690531],
       [-2.92381893, -1.52163862, -0.39970846],
       [-2.8677694 , -1.74044564, -0.49648329]])
```

Dataframe featuring the 3 principal components

In [91]:
```
PCA_dataset2 = pd.DataFrame(data = principalComponents2, columns = ['component3', 'compo
PCA_dataset2.head()
```

Loading [MathJax]/extensions/Safe.js

Out[91]:

	component3	component4	component5
0	9.289979	1.923575	4.542351
1	9.965293	1.923071	3.956595
2	9.741992	0.497274	3.398265
3	4.672389	1.970063	6.432226
4	8.480569	0.384073	2.500090

Extracting the three features

In [92]:

```
principal_component3 = PCA_dataset2['component3']
principal_component4 = PCA_dataset2['component4']
principal_component5 = PCA_dataset2['component5']
```

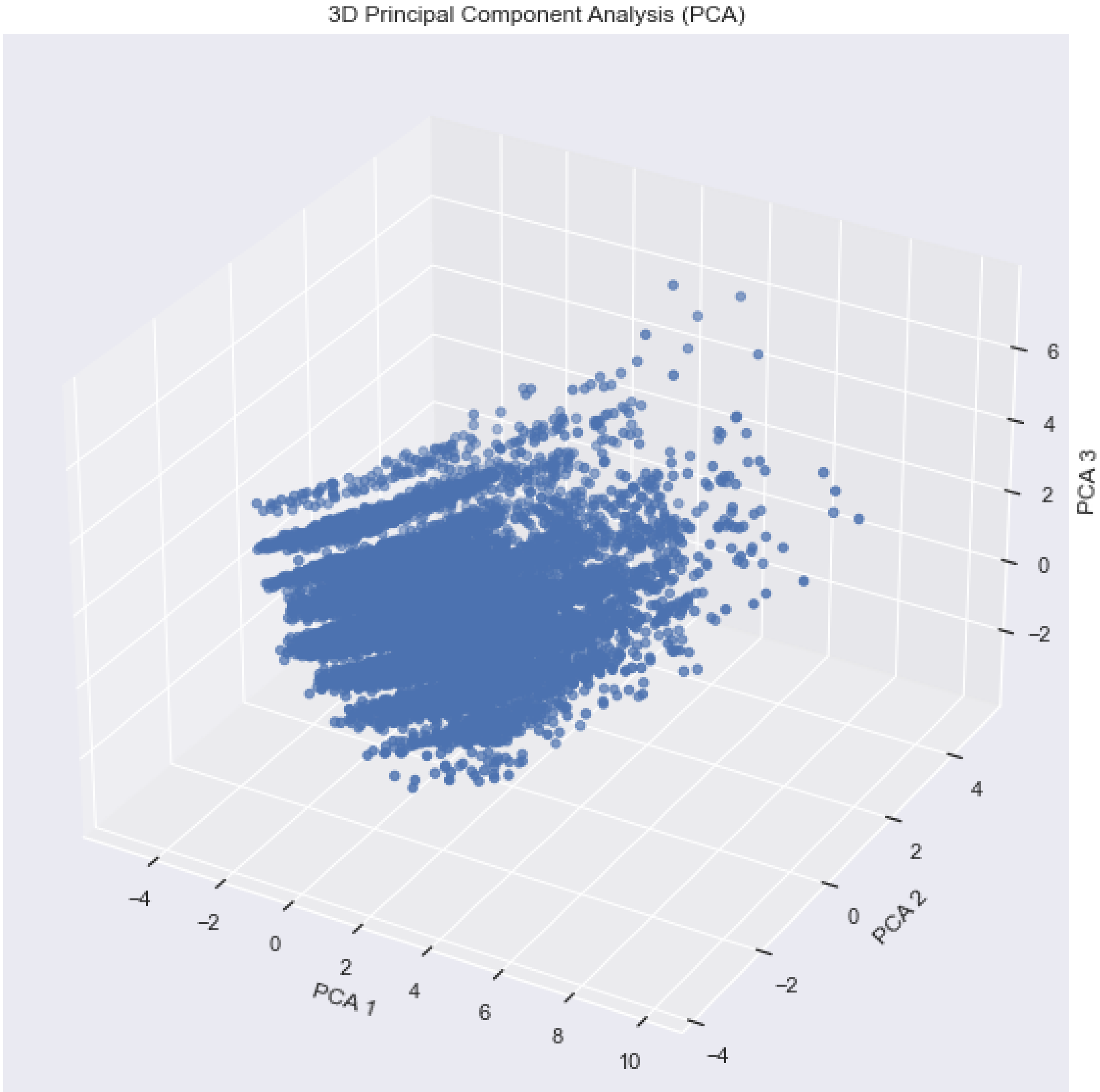
3D PCA

In [93]:

```
ax = plt.figure(figsize=(10,10)).gca(projection='3d')
plt.title('3D Principal Component Analysis (PCA)')
ax.scatter(
    xs=principal_component3,
    ys=principal_component4,
    zs=principal_component5,
    #c = x_kmeans
)
ax.set_xlabel('PCA 1')
ax.set_ylabel('PCA 2')
ax.set_zlabel('PCA 3')
plt.show()
```

```
C:\Users\richa\AppData\Local\Temp\ipykernel_19372\833820981.py:1: MatplotlibDeprecationW
arning:

Calling gca() with keyword arguments was deprecated in Matplotlib 3.4. Starting two mino
r releases later, gca() will take no keyword arguments. The gca() function should only b
e used to get the current axes, or if no axes exist, create new axes with default keywor
d arguments. To create a new axes with non-default arguments, use plt.axes() or plt.subp
lot().
```

Loading [MathJax]/extensions/Safe.js



K-Mean

K-Mean clustering algorithm

In [94]:
```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters = 100, init = 'k-means++', random_state = 1)
x_kmeans = kmeans.fit_predict(principalComponents2)
```

Adding 3 principal component features along with cluster features

In [95]:
```
df2['Principal Component 3'] = principal_component3
df2['Principal Component 4'] = principal_component4
df2['Principal Component 5'] = principal_component5
df2['Cluster2'] = x_kmeans
```

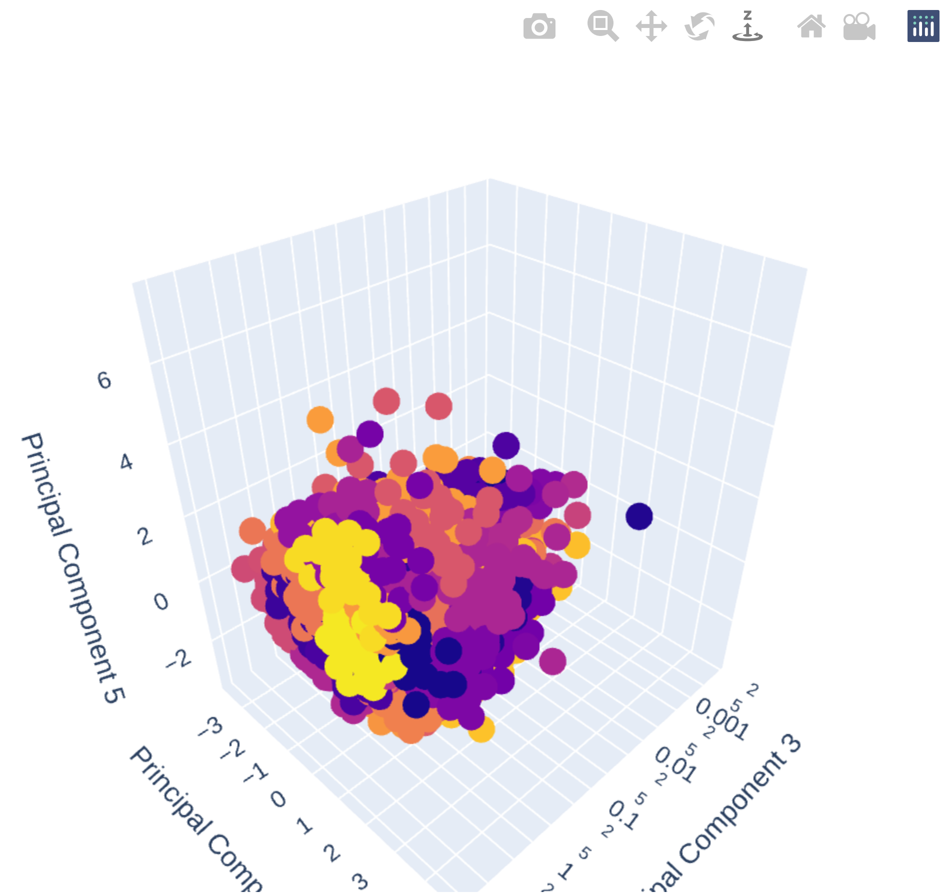
In [96]:
```
df2['Name'] = df['Name']
```

3D K-Mean

In [97]:
```
import plotly.express as px
fig = px.scatter_3d(df2, x='Principal Component 3', y='Principal Component 4', z='Princi
```

Loading [MathJax]/extensions/Safe.js

```
                     color=x_kmeans, log_x=True, hover_name="Name", hover_data=["Overall"])
fig.show()
```



Loading [MathJax]/extensions/Safe.js