As a user, I want to know overall reating so that I can undertand current FIFA progress

In [123]:
```
# Yihnew Eshetu
# yte9pc

from bs4 import BeautifulSoup
import requests
import pandas as pd


def connection():
    playersData = pd.DataFrame()
    for page in range(1,71):
        # Url address
        url = 'https://www.futbin.com/20/players?page='+str(page)+'&sort=Player_

        # Retrieve data from url
        response = requests.get(url)
        # BeautifulSoup parser
        soup = BeautifulSoup(response.text, 'html.parser')
        rows = soup.find_all('tr')

        playersInfo = []
        for row in range(len(rows)):

            if row not in range (0, 2):
                cols = rows[row].find_all('td')
                playerInfo = []

                for col in range(0,len(cols)):
                    if col == 0:
                        playersName = cols[col].text.strip()
                        playerInfo.append(playersName)
                        playerClubCountryLeague = cols[col].find('span', {'class
                        NoneTypeCheck(playerClubCountryLeague, playerInfo, 'a', 
                        NoneTypeCheck(playerClubCountryLeague, playerInfo, 'a', 
                        NoneTypeCheck(playerClubCountryLeague, playerInfo, 'a', 
                    elif col == 7:
                        playerInfo.append(cols[col].text.replace('\\', '/'))
                    elif col == 14:
                        playerInfo.append(cols[col].text.strip().split('cm')[0])
                    elif col not in range(3,5) and col != 15:
                        playerInfo.append(cols[col].text.strip())
                playersInfo.append(playerInfo)

        playersData = playersData.append(playersInfo, ignore_index = True)

    playersData.columns =  ['Name', 'Club', 'Country', 'League', 'Overall Rating
                            'Skill' , 'Weak Foot', 'Work Rate', 'Pace', 'Shootin
                            'Dribbling' , 'Defending', 'Physicality', 'Height',
                            'In Game Stats']
    playersData.to_csv('FIFA Player Info.csv')


def NoneTypeCheck(data, listname, item = None, iterator = None, get = None):
    if data is not None and get is not None:
        data = data.findAll(item)[iterator]
        data = data.get(get)
```

```
        listname.append(data)
    elif data is not None:
        listname.append(data.text)


if __name__ == '__main__':
    connection()
```

Start to do plots and analyze data

In [124]:
```
sample=pd.read_csv('FIFA Player Info.csv')
df=sample
```

In [125]:
```
#convert country names which are recongized by python
df['Country'] = pd.np.where(df['Country'] == "Holland", "Netherlands", df['Countr
df['Country'] = pd.np.where(df['Country'] == "England", "United Kingdom", df['Cou
```

In [126]:
```
#import modules for convert contry name to contry codes
import folium
import pycountry
df['Countryfullname'] = df['Country']
df['Countryfullname_cont'] = df['Country']
countries= df['Country'].unique().tolist()
print(countries)
```

```
['Argentina', 'Portugal', 'Brazil', 'Belgium', 'Slovenia', 'Germany', 'Egypt',
'Netherlands', 'Croatia', 'Italy', 'Spain', 'Uruguay', 'France', 'Poland', 'Uni
ted Kingdom', 'Senegal', 'Denmark', 'Gabon', 'Korea Republic', 'Costa Rica', 'B
osnia and Herzegovina', 'Slovakia', 'Colombia', 'Austria', 'Scotland', 'Greec
e', 'Serbia', 'Morocco', 'Sweden', 'Wales', 'Hungary', 'Switzerland', 'Algeri
a', 'Chile', 'Czech Republic', "Côte d'Ivoire", 'Mexico', 'Norway', 'Iceland',
'Finland', 'Togo', 'Montenegro', 'Ukraine', 'Russia', 'Guinea', 'Jamaica', 'Cam
eroon', 'Congo DR', 'Ghana', 'Albania', 'Venezuela', 'Armenia', 'Central Africa
n Republic', 'Israel', 'Nigeria', 'Australia', 'Mali', 'Romania', 'Japan', 'Tur
key', 'Paraguay', 'Northern Ireland', 'Cape Verde Islands', 'Tanzania', 'China
PR', 'Kosovo', 'Republic of Ireland', 'Tunisia', 'United States', 'Dominican Re
public', 'Burkina Faso', 'Syria', 'Peru', 'FYR Macedonia', 'Angola', 'South Afr
ica', 'Ecuador', 'Kenya', 'New Zealand', 'Equatorial Guinea', 'Gambia', 'Canad
a', 'Benin', 'Georgia', 'Estonia', 'Mozambique', 'Zimbabwe', 'Uzbekistan', 'Cub
a', 'Iraq', 'Honduras', 'Guinea-Bissau', 'Cyprus', 'Madagascar', 'Moldova', 'Ph
ilippines', 'Iran']
```

In [127]:
```
#convert country to country code to three digits
import pycountry

input_countries = df['Country']

countries = {}
for country in pycountry.countries:
    countries[country.name] = country.alpha_3

codes = [countries.get(country, 'Unknown code') for country in input_countries]
```

In [128]:
```
#convert country to country code to two digits then continent
import pycountry
import pycountry_convert as pc
input_countries = df['Countryfullname_cont']

countries = {}
for country in pycountry.countries:
    countries[country.name] = country.alpha_2

codes_cont = [countries.get(country, 'Unknown code') for country in input_countr
```

In [129]:
```
df['Country'] = codes
df.head(5)
```

Out[129]:

	Unnamed: 0	Name	Club	Country	League	Overall Rating	Position	Skill	Weak Foot	Work Rate	...
0	0	Lionel Messi	FC Barcelona	ARG	LaLiga Santander	94	RW	4	4	M / L	...
1	1	Cristiano Ronaldo	Piemonte Calcio	PRT	Serie A TIM	93	ST	5	4	H / L	...
2	2	Neymar Jr	Paris Saint-Germain	BRA	Ligue 1 Conforama	92	LW	5	5	H / M	...
3	3	Kevin De Bruyne	Manchester City	BEL	Premier League	91	CAM	4	5	H / H	...
4	4	Eden Hazard	Real Madrid	BEL	LaLiga Santander	91	LW	4	4	H / M	...

5 rows × 21 columns

In [130]:
```
import pycountry_convert as pc
df['Countryfullname_cont1'] = codes_cont
df.head(5)
```

Out[130]:

	Unnamed: 0	Name	Club	Country	League	Overall Rating	Position	Skill	Weak Foot	Work Rate	...
0	0	Lionel Messi	FC Barcelona	ARG	LaLiga Santander	94	RW	4	4	M / L	...
1	1	Cristiano Ronaldo	Piemonte Calcio	PRT	Serie A TIM	93	ST	5	4	H / L	...
2	2	Neymar Jr	Paris Saint-Germain	BRA	Ligue 1 Conforama	92	LW	5	5	H / M	...
3	3	Kevin De Bruyne	Manchester City	BEL	Premier League	91	CAM	4	5	H / H	...
4	4	Eden Hazard	Real Madrid	BEL	LaLiga Santander	91	LW	4	4	H / M	...

5 rows × 22 columns

In [131]:
```
df.drop(df.loc[df['Countryfullname_cont1']=='Unknown code'].index, inplace=True)
```

In [132]:
```
#Convery contry code to continent
from pycountry_convert import country_alpha2_to_continent_code, country_name_to_
cont1s=df['Countryfullname_cont1'].tolist()
change_cont1s = []
for cont1 in cont1s:
    continent_name = country_alpha2_to_continent_code(cont1)
    change_cont1s.append(continent_name)
df['Countryfullname_cont']=change_cont1s
df.tail(10)
```

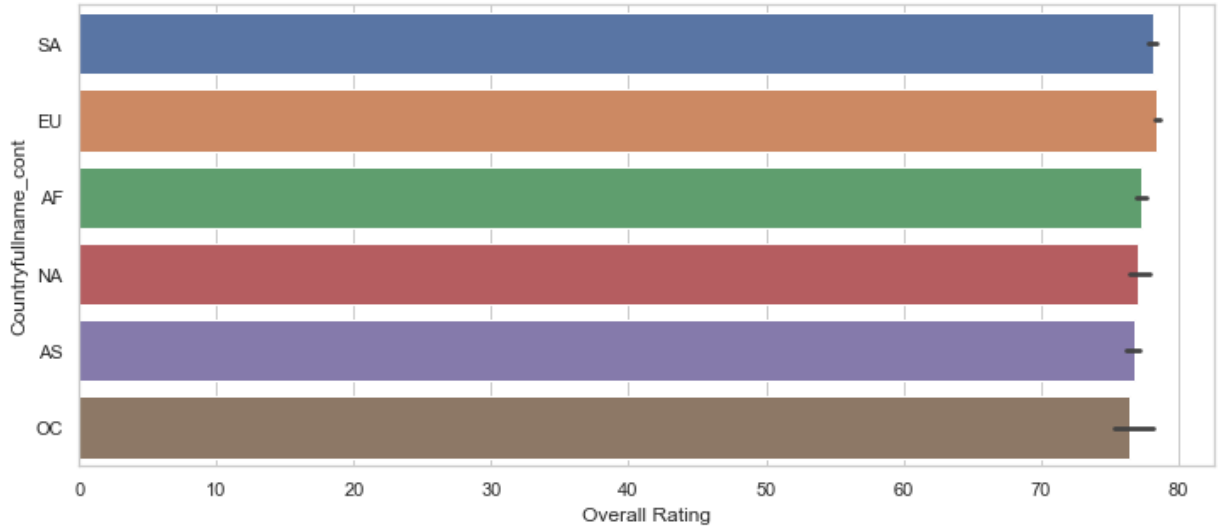
Out[132]:

	Unnamed: 0	Name	Club	Country	League	Overall Rating	Position	Skill	Weak Foot	Work Rate	...
2077	2077	Jesé	Sporting CP	ESP	Liga NOS	75	LW	4	4	M / M	...
2078	2078	Lucas Lima	Al Ahli	BRA	Saudi Professional League	75	LB	3	1	H / M	...
2079	2079	Lisandro Magallán	D. Alavés	ARG	LaLiga Santander	75	CB	2	3	L / M	...
2080	2080	Éder Balanta	Club Brugge KV	COL	Belgium Pro League	75	CDM	2	2	M / H	...
2081	2081	Guido Carrillo	CD Leganés	ARG	LaLiga Santander	75	ST	3	3	M / M	...
2082	2082	Jason	Getafe CF	ESP	LaLiga Santander	75	RM	3	4	M / L	...
2083	2083	Kenedy	Getafe CF	BRA	LaLiga Santander	75	LM	4	2	H / H	...
2084	2084	Wesley Hoedt	Royal Antwerp FC	NLD	Belgium Pro League	75	CB	2	3	M / H	...
2085	2085	Marius Wolf	Hertha BSC	DEU	Bundesliga	75	RB	3	4	H / M	...
2086	2086	Adam Ounas	OGC Nice	DZA	Ligue 1 Conforama	75	LM	4	4	M / L	...

10 rows × 22 columns

In [158]:
```
# >>> import seaborn as sns
# >>> sns.set(style="whitegrid")
# >>> tips = sns.load_dataset("tips")
# >>> ax = sns.barplot(x="day", y="total_bill", data=tips)
```

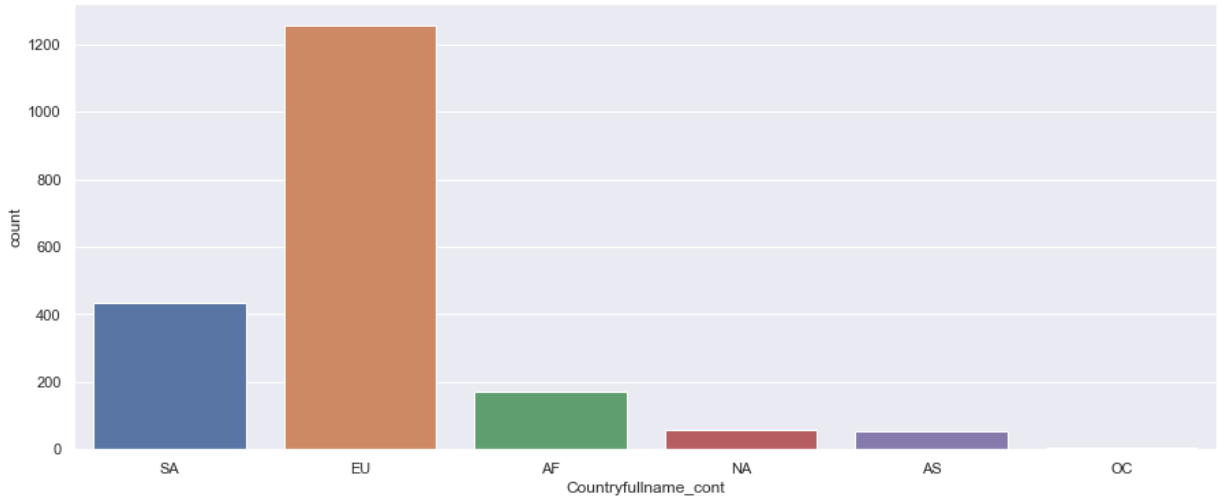
In [157]:
```
import seaborn as sns
ax = sns.barplot(x="Overall Rating", y="Countryfullname_cont", data=sample)
```



continents, NA = "North America", SA = "South Amierica", AS = "Asia", OC = "Australia", AF = "Africa"

In [135]:
```
# Sort players based on their countries
plt.figure(figsize=(15,6))
sns.countplot(x="Countryfullname_cont", data=sample)
```

Out[135]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x15123431cc0>
```



In [136]:
```
df2=df.head(100)
df.describe()
```

Out[136]:

	Unnamed: 0	Overall Rating	Skill	Weak Foot	Pace	Shooting	Passing
count	1981.000000	1981.000000	1981.000000	1981.000000	1981.000000	1981.000000	1981.000000
mean	1033.047956	78.204947	2.923271	3.218576	71.536093	65.774861	69.970722
std	602.780522	3.276238	0.964610	0.726259	11.852986	13.372850	8.419623
min	0.000000	75.000000	1.000000	1.000000	29.000000	20.000000	38.000000
25%	507.000000	76.000000	2.000000	3.000000	65.000000	60.000000	66.000000
50%	1030.000000	77.000000	3.000000	3.000000	73.000000	70.000000	71.000000
75%	1552.000000	80.000000	4.000000	4.000000	80.000000	75.000000	76.000000
max	2086.000000	94.000000	5.000000	5.000000	96.000000	93.000000	93.000000

In [137]:
```
#As a user, I want to know players from which continents
df.groupby('Countryfullname_cont')['Name'].count()
```

Out[137]:
```
Countryfullname_cont
AF     171
AS      55
EU    1257
NA      57
OC       6
SA     435
Name: Name, dtype: int64
```

In [138]:
```
df.groupby('Countryfullname')['Name'].count()
```

Out[138]:
```
Countryfullname
Albania             3
Algeria            14
Angola              3
Argentina         165
Armenia             2
                 ...
United Kingdom    120
United States      13
Uruguay            33
Uzbekistan          1
Zimbabwe            2
Name: Name, Length: 80, dtype: int64
```

In [139]:
```
#As a user, I want to know player from which countries
#df.groupby('Countryfullname')['Name'].count()
df.groupby('Name', as_index=False).agg({"Countryfullname_cont": "sum"})
```

Out[139]:

	Name	Countryfullname_cont
0	Aaron Cresswell	EU
1	Aaron Hunt	EU
2	Aaron Long	NA
3	Aaron Mooy	OC
4	Aaron Wan-Bissaka	EU
...	...	...
1894	Łukasz Fabiański	EU
1895	Łukasz Piszczek	EU
1896	Łukasz Skorupski	EU
1897	Łukasz Teodorczyk	EU
1898	Šime Vrsaljko	EU

1899 rows × 2 columns

In [140]:
```
# grouped = df.groupby('Countryfullname_cont').agg("Overall Rating": [min, max, r
# # Using ravel, and a string join, we can create better names for the columns:
# grouped.columns = ["_".join(x) for x in grouped.columns.ravel()]
df.groupby('Countryfullname_cont', as_index=False).agg({"Overall Rating": "sum"}
```

Out[140]:

	Countryfullname_cont	Overall Rating
0	AF	13221
1	AS	4222
2	EU	98643
3	NA	4395
4	OC	459
5	SA	33984

In [141]:
```
df.groupby('Countryfullname', as_index=False).agg({"Overall Rating": "sum"})
```

Out[141]:

	Countryfullname	Overall Rating
0	Albania	238
1	Algeria	1091
2	Angola	230
3	Argentina	12835
4	Armenia	162
...	...	...
75	United Kingdom	9326
76	United States	990
77	Uruguay	2605
78	Uzbekistan	75
79	Zimbabwe	151

80 rows × 2 columns

In [142]:
```
grouped_continent = df.groupby('Countryfullname_cont').agg({'Overall Rating': ['
print(grouped_continent)
```

```
                    Overall Rating
                              mean min max
Countryfullname_cont
AF                        77.315789  75  90
AS                        76.763636  75  81
EU                        78.474940  75  93
NA                        77.105263  75  87
OC                        76.500000  75  80
SA                        78.124138  75  94
```

In [143]:
```
grouped_continent = df.groupby('Countryfullname').agg({'Overall Rating': ['mean'
print(grouped_continent)
```

```
                Overall Rating
                          mean min max
Countryfullname
Albania               79.333333  76  82
Algeria               77.928571  75  84
Angola                76.666667  75  78
Argentina             77.787879  75  94
Armenia               81.000000  81  81
...                         ...  ..  ..
United Kingdom        77.716667  75  89
United States         76.153846  75  79
Uruguay               78.939394  75  89
Uzbekistan            75.000000  75  75
Zimbabwe              75.500000  75  76

[80 rows x 3 columns]
```

In [144]:
```
df[df['Position'] == 'LW'].groupby('Countryfullname_cont').agg(
    # Get max of the duration column for each group
    max_Overall_Rating=('Overall Rating', max),
    # Get min of the duration column for each group
    min_Overall_Rating=('Overall Rating', min),
    # Get sum of the duration column for each group
    sum_Overall_Rating=('Overall Rating', sum),
    # Apply a lambda to date column
    #num_days=("date", lambda x: (max(x) - min(x)).days)
)
```

Out[144]:

Countryfullname_cont	max_Overall_Rating	min_Overall_Rating	sum_Overall_Rating
AF	88	75	548
AS	80	80	80
EU	91	75	2341
NA	82	79	161
SA	92	76	1200

In [159]:
```
df.groupby('Countryfullname_cont')[['Overall Rating']].mean()
```

Out[159]:

Countryfullname_cont	Overall Rating
AF	77.315789
AS	76.763636
EU	78.474940
NA	77.105263
OC	76.500000
SA	78.124138

In [160]:
```
country_geo = 'world-countries.json'
```

In [161]:
```
stage = df
data_to_plot = stage[['Country','Overall Rating']]
```

In [162]:
```
stage = df
data_to_plot = stage[['Country','Overall Rating']]
```

In [163]:
```
hist_indicator =  'Overall Rating'
```

In [164]:
```
data_to_plot.head()
```

Out[164]:

	Country	Overall Rating
0	ARG	94
1	PRT	93
2	BRA	92
3	BEL	91
4	BEL	91

In [165]:
```
import os
os.getcwd()
```

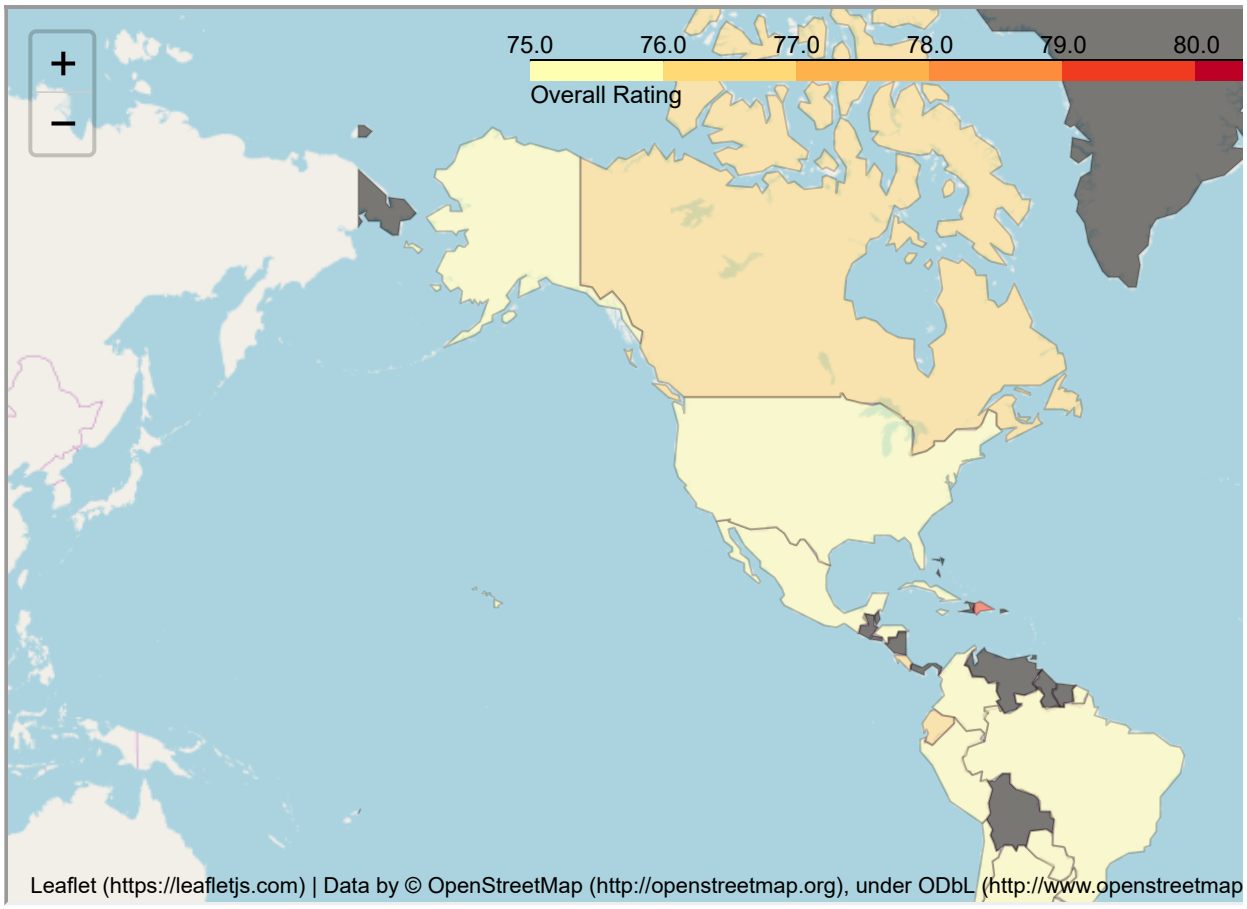
Out[165]:
```
'C:\\Users\\gladies\\CS5010_semesterProject'
```

In [166]:
```
import json
country_geo = json.load(open("world-countries.json"))
#country_geo = 'world-countries.json'
map = folium.Map(location=[100, 0], zoom_start=1.5)
map.choropleth(geo_data=country_geo, data=data_to_plot,
             columns=['Country', 'Overall Rating'],
             key_on='feature.id',
             fill_color='YlOrRd', fill_opacity=0.5, line_opacity=0.2,
             legend_name=hist_indicator)
```

In [167]:
```
# Create Folium plot
map.save('plot_data.html')
```

In [168]:
```
# Create Folium plot
map.save('plot_data.html')
# Import the Folium interactive html file
from IPython.display import HTML
HTML('<iframe src=plot_data.html width=700 height=450></iframe>')
```

Out[168]:



In [169]:
```
df1=df
stage = df1
data_to_plot = stage[['Country','Skill']]
```

In [170]:
```
hist_indicator =  'Skill'
```

In [171]:
```
data_to_plot.head()
```

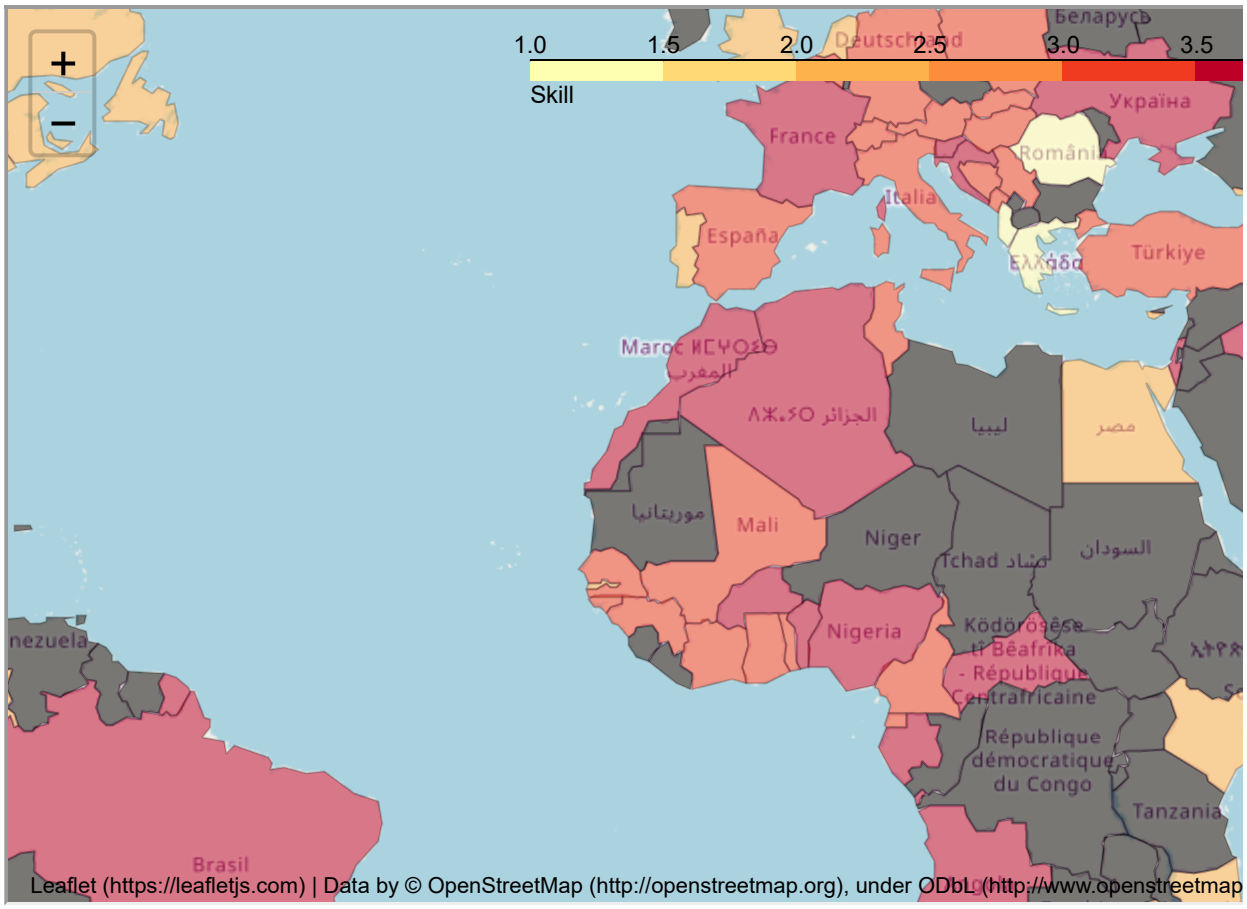
Out[171]:

	Country	Skill
0	ARG	4
1	PRT	5
2	BRA	5
3	BEL	4
4	BEL	4

In [172]:
```
import json
country_geo = json.load(open("world-countries.json"))
#country_geo = 'world-countries.json'
map = folium.Map(location=[100, 0], zoom_start=1.5)
map.choropleth(geo_data=country_geo, data=data_to_plot,
             columns=['Country', 'Skill'],
             key_on='feature.id',
             fill_color='YlOrRd', fill_opacity=0.5, line_opacity=0.2,
             legend_name=hist_indicator)
```

In [173]:
```
# Create Folium plot
map.save('plot_data1.html')
# Import the Folium interactive html file
from IPython.display import HTML
HTML('<iframe src=plot_data1.html width=700 height=450></iframe>')
```

Out[173]:



In [*]:
```
df= sns.PairGrid(sample)
df.map_diag(sns.kdeplot)
df.map_offdiag(sns.kdeplot, n_levels=6);
```

In [*]:
```
import seaborn as sns;
# Scatterplot arguments
sns.lmplot(x='Overall Rating', y='Physicality', data=sample,
           fit_reg=False, # No regression line
           hue='Position')   # Color by evolution stage
```

In [ ]:
```
# Scatterplot arguments
sns.lmplot(x='Overall Rating', y='Dribbling', data=sample,
           fit_reg=False, # No regression line
           hue='Position')   # Color by evolution stage
```

In [ ]:
```
# Scatterplot arguments
sns.lmplot(x='Overall Rating', y='Defending', data=sample,
           fit_reg=False, # No regression line
           hue='Position')   # Color by evolution stage
```

In [ ]:
```
# Scatterplot arguments
sns.lmplot(x='Overall Rating', y='Height', data=sample,
           fit_reg=False, # No regression line
           hue='Position')   # Color by evolution stage
```

In [ ]:
```
# Scatterplot arguments
sns.lmplot(x='Overall Rating', y='Skill', data=sample,
           fit_reg=False, # No regression line
           hue='Position')   # Color by evolution stage
```

In [ ]:
```
# Scatterplot arguments
sns.lmplot(x='Overall Rating', y='Base Stats', data=sample,
           fit_reg=False, # No regression line
           hue='Position')   # Color by evolution stage
```

In [ ]:
```
# Scatterplot arguments
sns.lmplot(x='Overall Rating', y='In Game Stats', data=sample,
           fit_reg=False, # No regression line
           hue='Position')   # Color by evolution stage
```

In [ ]:
```
plt.figure(figsize=(15,6))
sns.countplot(x="Overall Rating",data=sample)
```

In [ ]:
```
df = sns.jointplot(x="Overall Rating", y="Physicality", data=sample, kind="kde",
df.plot_joint(plt.scatter, c="w", s=30, linewidth=1, marker="+")
df.ax_joint.collections[0].set_alpha(0)
df.set_axis_labels("Overall Rating", "Physically");
```

In [ ]:
```
df = sns.jointplot(x="Overall Rating", y="Shooting", data=sample, kind="kde", co
df.plot_joint(plt.scatter, c="w", s=30, linewidth=1, marker="+")
df.ax_joint.collections[0].set_alpha(0)
df.set_axis_labels("Overall Rating", "Shooting");
```

In [ ]:
```
# Generate sequential data and plot
plt.figure(figsize=(15,6))
sd = sample.sort_values('Overall Rating', ascending=False)[:8]
x1 = np.array(list(sd['Name']))
y1 = np.array(list(sd['Overall Rating']))
sns.barplot(x1, y1, palette= "colorblind")
plt.ylabel("Overall Rating")
```

In [ ]:
```
# Generate sequential data and plot
plt.figure(figsize=(15,6))
sd = sample.sort_values('Overall Rating', ascending=False)[:10]
x1 = np.array(list(sd['Country']))
y1 = np.array(list(sd['Overall Rating']))
sns.barplot(x1, y1, palette= "colorblind")
plt.ylabel("Overall Rating")
```

In [ ]:
```
# Generate sequential data and plot
plt.figure(figsize=(15,6))
sd = sample.sort_values('Overall Rating', ascending=False)[:10]
x1 = np.array(list(sd['Countryfullname']))
y1 = np.array(list(sd['Overall Rating']))
sns.barplot(x1, y1, palette= "colorblind")
plt.ylabel("Overall Rating")
```

In [ ]:
```
f_fuko = sample.loc[sample['Position']=='RW']['Overall Rating']
m_fuko = sample.loc[sample['Position']=='ST']['Overall Rating']
sns.distplot(f_fuko, hist=True, kde=True, rug=False, hist_kws={'edgecolor':'blacl
sns.distplot(m_fuko, hist=True, kde=True, rug=False, hist_kws={'edgecolor':'blacl
plt.legend()
```

In [ ]:
```
g = sns.pairplot(sample, hue="Position")
```

In [ ]:
```
x = sample['Overall Rating']
ax = sns.distplot(x, hist=True, kde=True, rug=False, color='m', bins=25, hist_kws
plt.show()
```

In [ ]:
```
from matplotlib.pyplot import figure
figure(num=None, figsize=(12, 5), dpi=80, facecolor='w', edgecolor='k')
import seaborn as sns; sns.set()
import matplotlib.pyplot as plt
ax = sns.scatterplot(x="Shooting", y='Overall Rating',  size="Position", data=sa
sns.regplot(x="Shooting", y="Overall Rating", data=sample)
```

In [ ]:
```
g = sns.pairplot(sample, vars=["Overall Rating", "Skill"], hue="Position")
```

In [ ]:
```
g = sns.pairplot(sample, vars=["Overall Rating", "Pace", "Shooting", "Passing",
```

In [ ]:
```
g = sns.pairplot(sample, vars=["Overall Rating", "Pace", "Shooting", "Passing",
```

In [ ]:
```
g = sns.pairplot(sample, vars=["Overall Rating", "Pace", "Shooting", "Passing",
```

In [ ]:
```
g = sns.pairplot(sample, vars=["Overall Rating", "Height", "Base Stats", "In Game
```

Sort players based on their countries

In [ ]:
```
#sort players based on their countries
sample.sort_values(by=['Countryfullname'], inplace=True)
```

In [ ]:
```

```