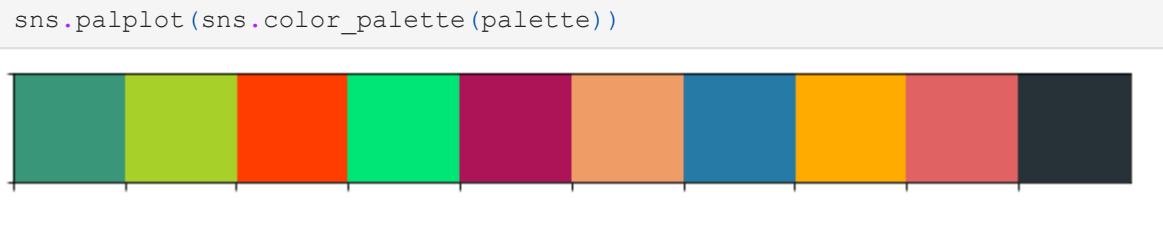
In [1]:
```
#### Libraries that we are going to use

import matplotlib
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from matplotlib.offsetbox import (OffsetImage,AnnotationBbox)
import pandas as pd
import numpy as np
from math import pi
import requests
import seaborn as sns
from time import time
from datetime import datetime, timedelta
import matplotlib.patches as mpatches
from IPython.display import Image
%matplotlib inline
```

In [4]:
```
#### I create a palette of colors to use in graphics

palette = ["#3a9679", "#a7d129","#ff3d00", "#00e676", "#ad1457", "#f09c67", "#257aa6",
           "#ffab00", "#e16262", "#263238"]
sns.palplot(sns.color_palette(palette))
```



In [5]:
```
#### Read the file

players = pd.read_csv('data.csv')
```

In [7]:
```
#### These data have a strange format so we clean them

players[['Value', 'Wage']].head(1)
```

Out[7]:

	Value	Wage
0	€110.5M	€565K

In [8]:
```
####  use this function to clean the values and salaries since they are in M and K

def clear(x):
    try:
        if 'M' in x:
            return float(x.split("M")[0][1:])
        elif 'K' in x:
            return float(x.split("K")[0][1:])/1000
    except:
        return 0

players['Value'] = players['Value'].apply(lambda x : clear(x))
players['Wage'] = players['Wage'].apply(lambda x : clear(x))

players[['Value', 'Wage']].head(1)
```

Out[8]:

	Value	Wage
0	110.5	0.565

In [9]:
```
#### Some column names are separated so let's put them together

players.rename(columns={'Club Logo':'Club_Logo',
                        'Preferred Foot' : 'Preferred_Foot', 'Body Type' : 'Body_Type',
                        'Jersey Number' : 'Jersey_Number', 'Loaned From' : 'Loaned_From',
                        'Contract Valid Until' : 'Contract_Valid',
                        'Release Clause' : 'Release_Clause'}, inplace=True)
```

In [10]:
```
#### least important ones with No Data and 0 depending on the type of data that they are

players = players.dropna(subset=['Club', 'Position', 'Value', 'Wage'])
players[['Release_Clause','Loaned_From','Joined']] = players[['Release_Clause',
                                                              'Loaned_From',
                                                              'Joined']].fillna('No data')
players = players.fillna(0)
players = players.reset_index()
```

In [11]:
```
#### we check for null data

players.isnull().sum().sum()
```

Out[11]: 0

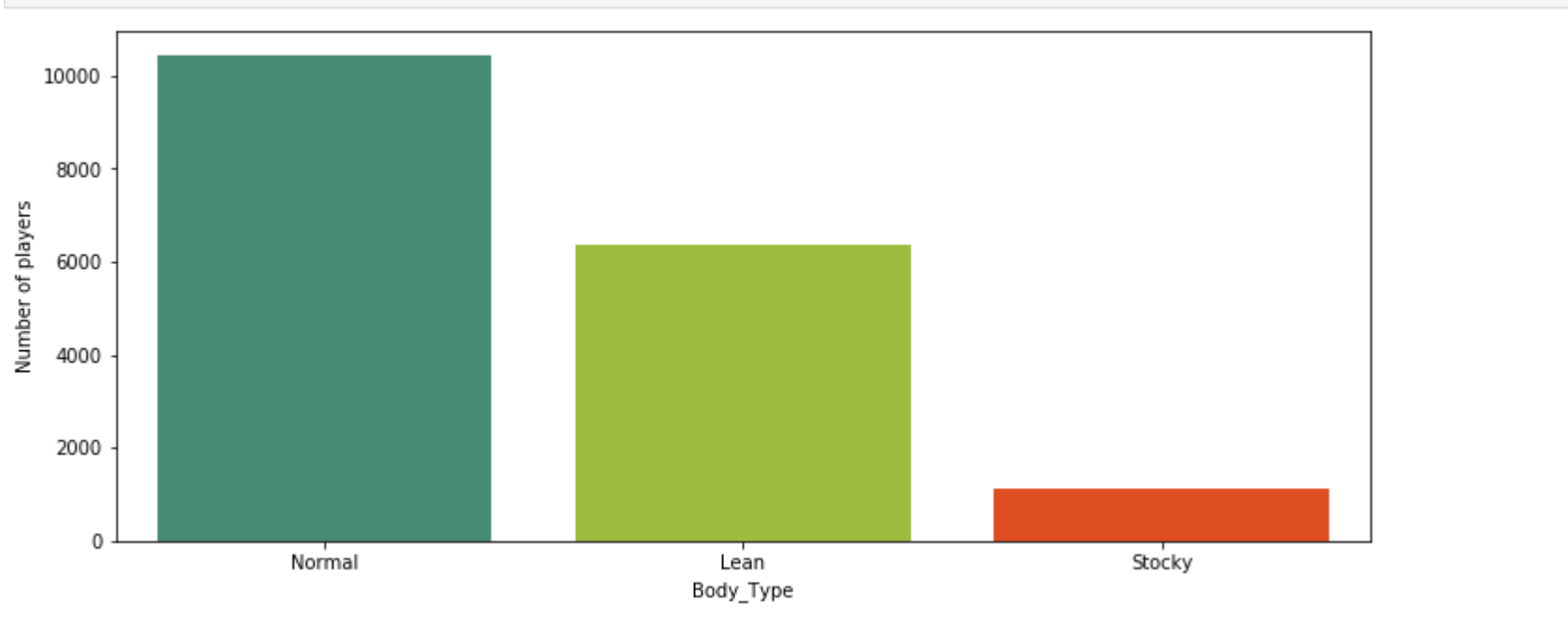
In [12]:
```
#### Another data that is wrong is the type of body, in a few cases the information is missing
#### and completed with the name or other data, we will create a function to solve this

players.Body_Type.unique()
```

Out[12]:
```
array(['Messi', 'C. Ronaldo', 'Neymar', 'Lean', 'Normal', 'Courtois',
       'Stocky', 'PLAYER_BODY_TYPE_25', 'Shaqiri', 'Akinfenwa'],
      dtype=object)
```

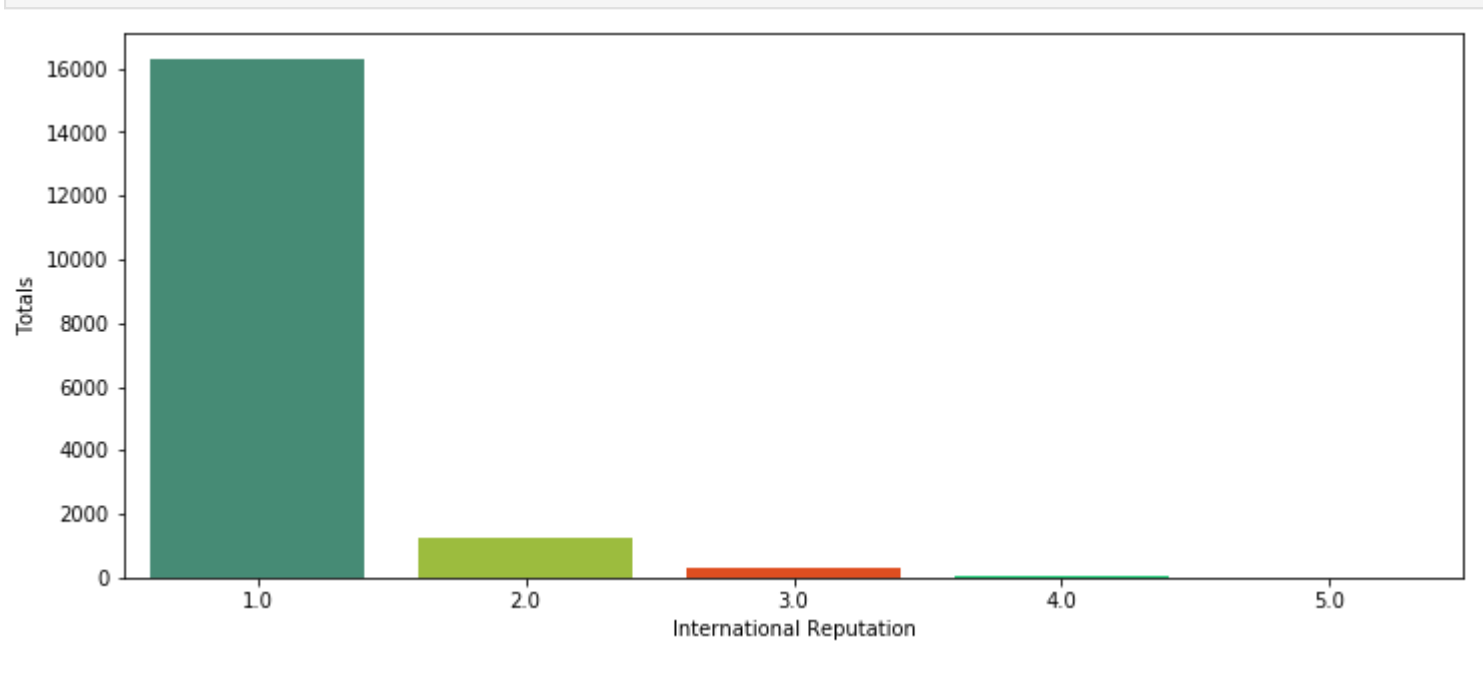
In [13]:
```
def body_type_func(players):
    if players in('Messi', 'C. Ronaldo', 'Neymar', 'Courtois', 'PLAYER_BODY_TYPE_25',
                  'Shaqiri', 'Akinfenwa'):
        return 'Normal'
    else:
        return players

players['Body_Type'] = players['Body_Type'].apply(body_type_func)
```

Some basic graphics

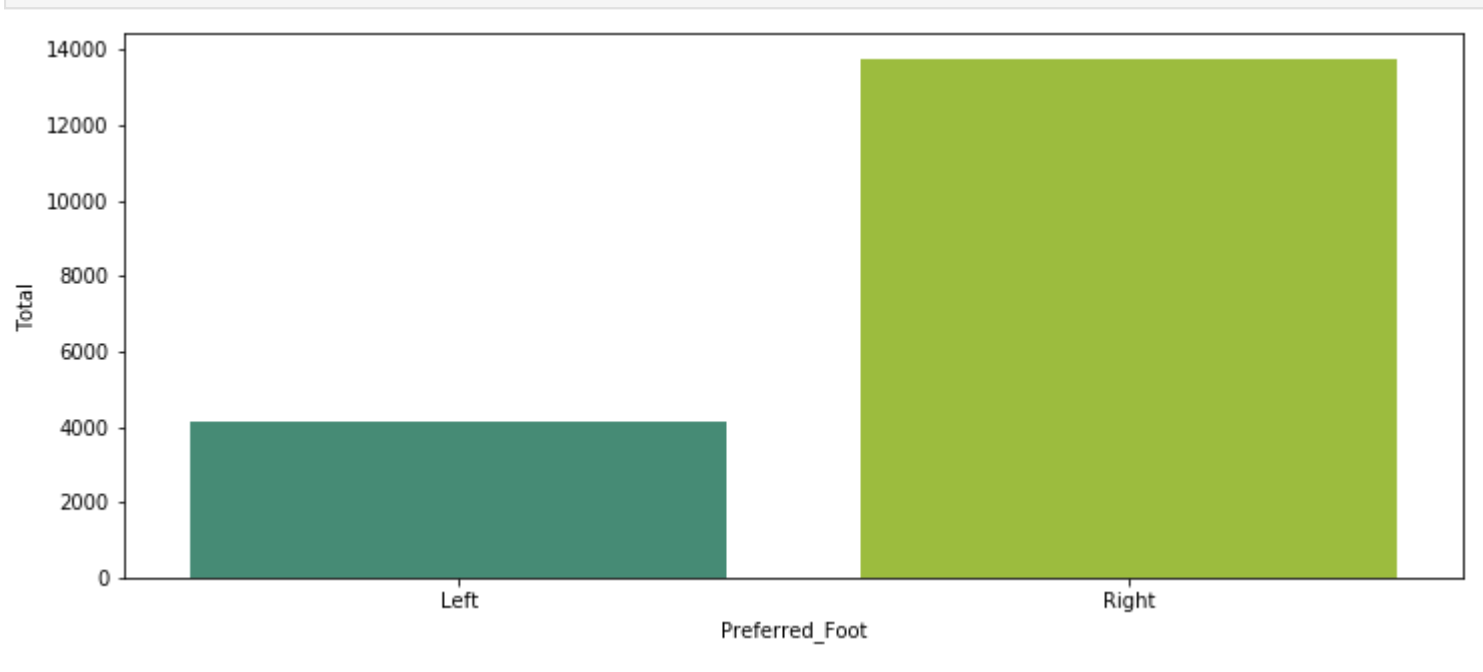
In [14]:
```
plt.figure(figsize=(12,5))
ax = sns.countplot('Body_Type', data = players,
                   palette= palette)
plt.ylabel('Number of players')
plt.show()
```



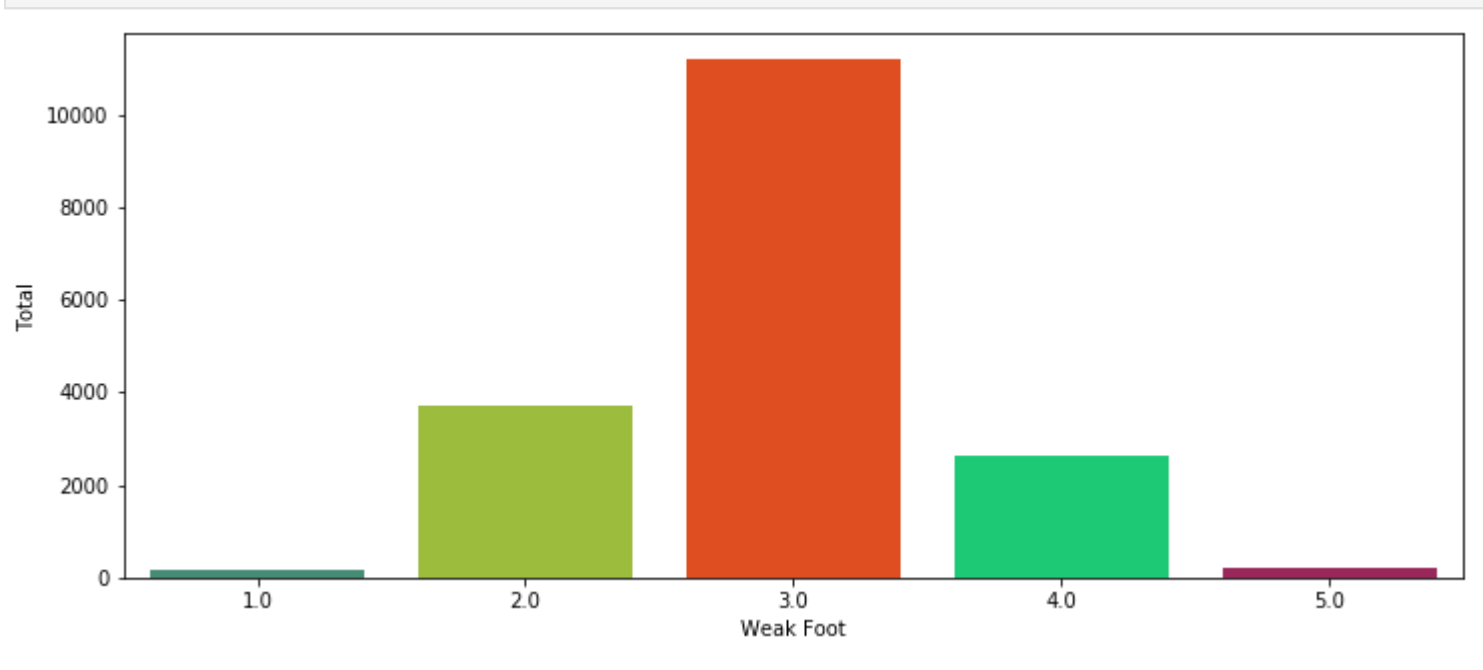
In [15]:
```
plt.figure(figsize=(12,5))
ax = sns.countplot('International Reputation', data = players,
                   palette= palette)
plt.ylabel('Totals')
plt.show()
```



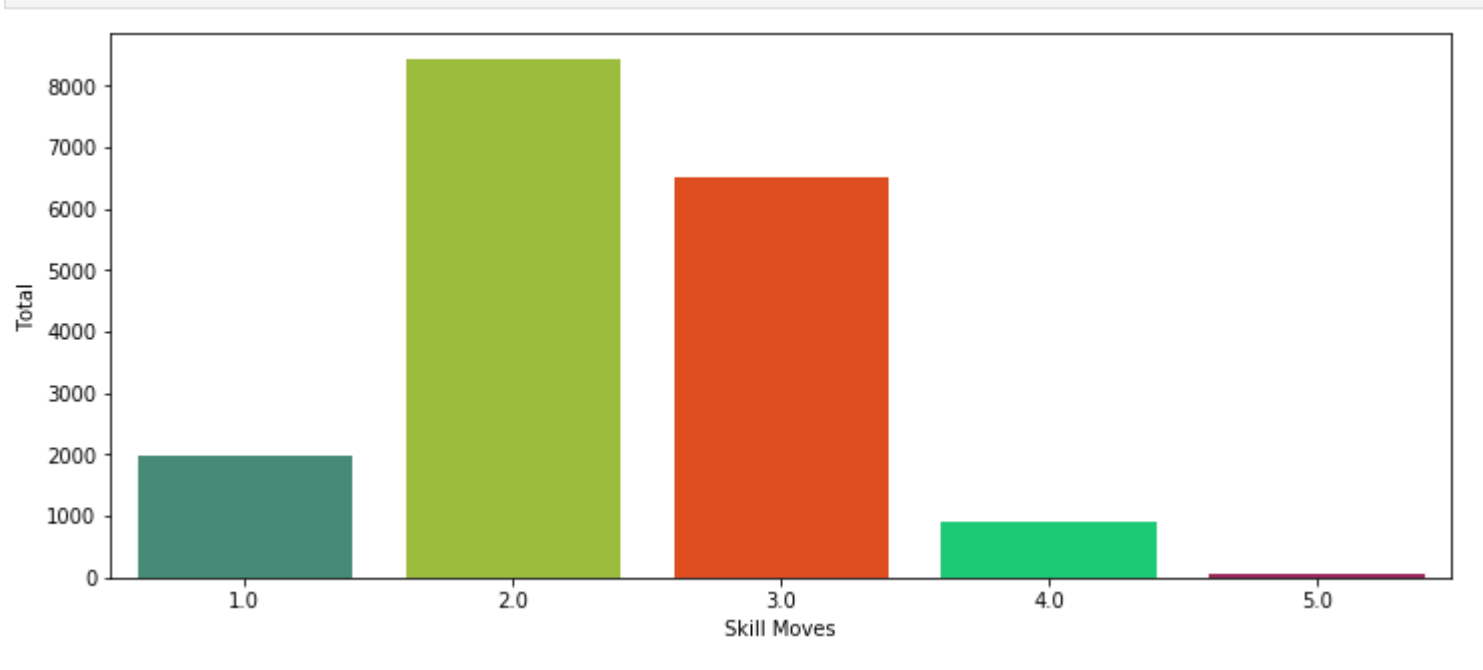
In [16]:
```
plt.figure(figsize=(12,5))
ax = sns.countplot('Preferred_Foot', data = players,
                   palette= palette)
plt.ylabel('Total')
plt.show()
```



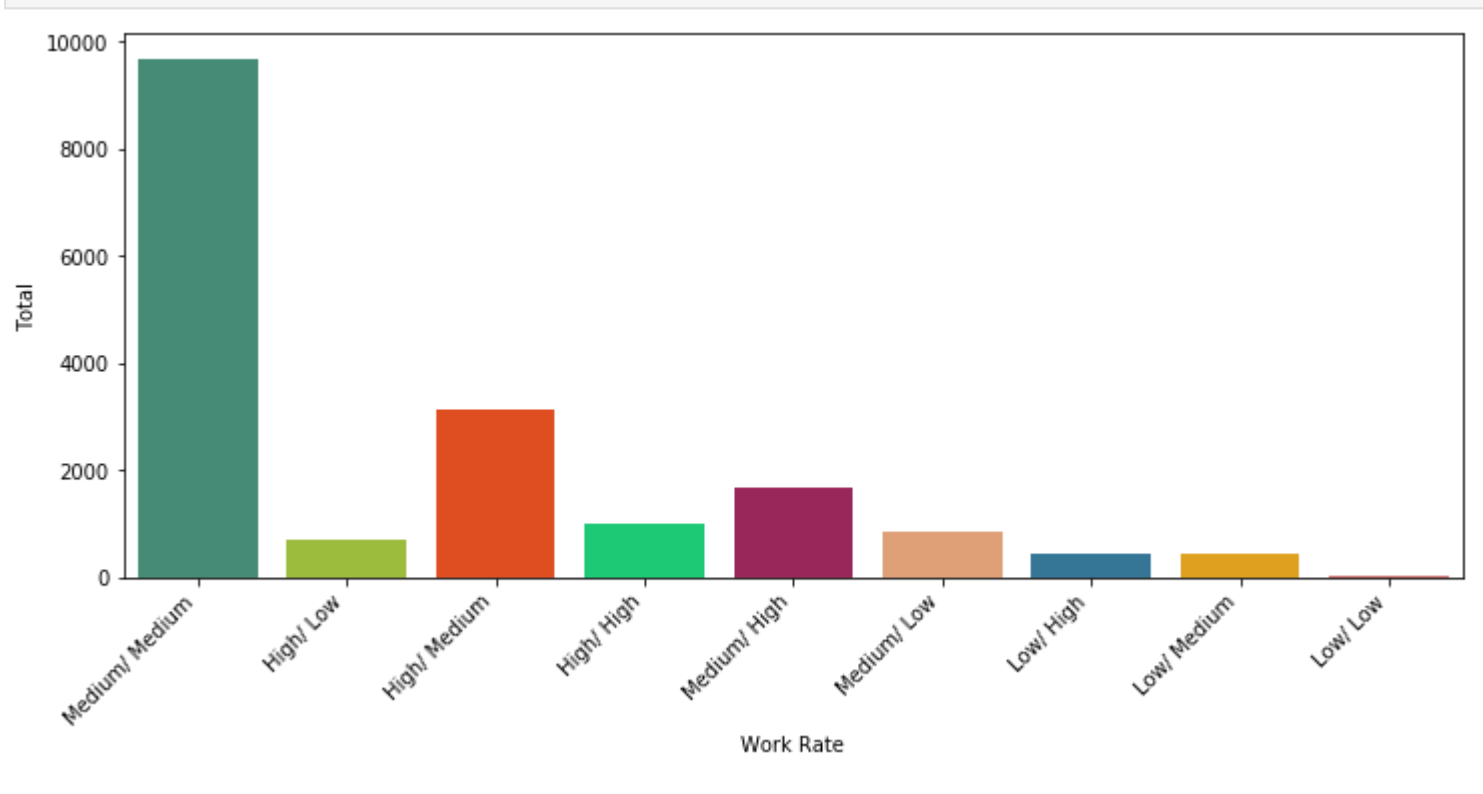
In [17]:
```
plt.figure(figsize=(12,5))
ax = sns.countplot('Weak Foot', data = players,
                   palette= palette)
plt.ylabel('Total')
plt.show()
```



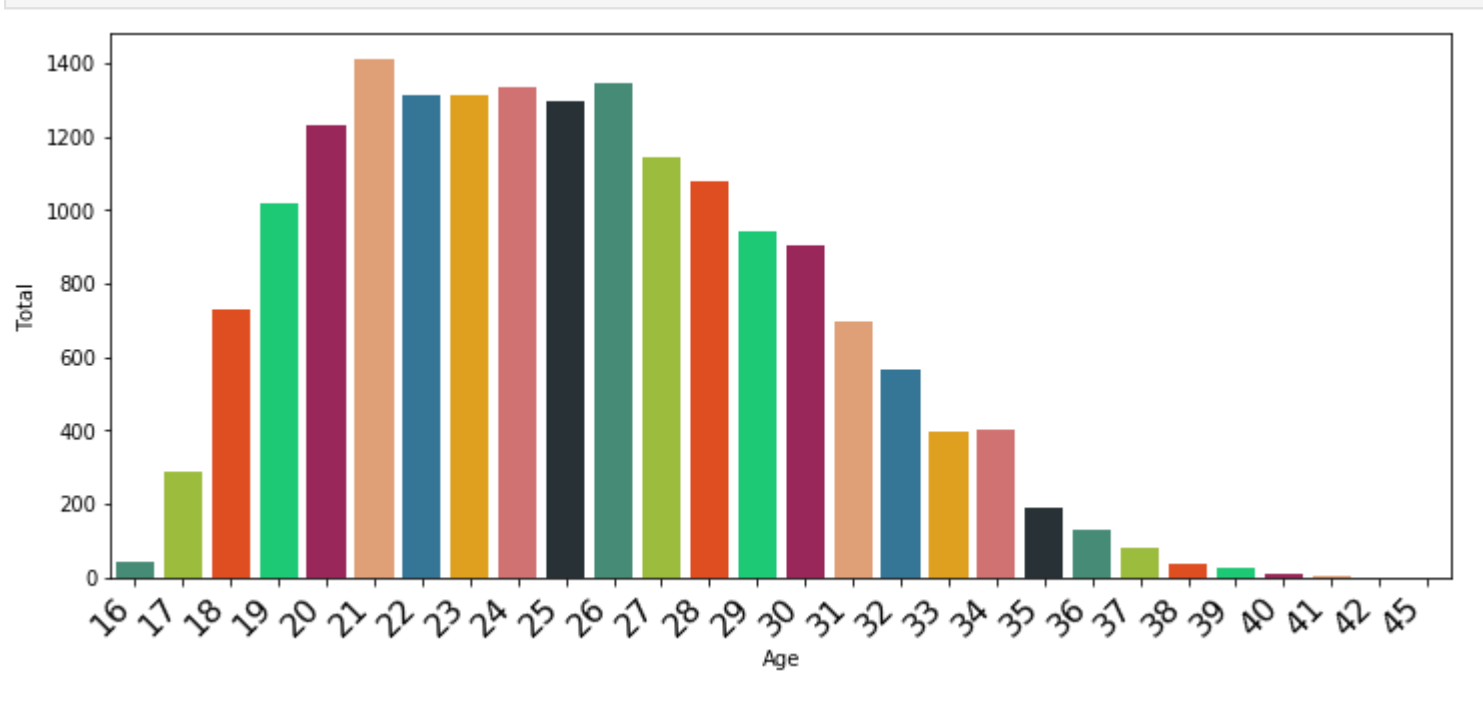
In [18]:
```
plt.figure(figsize=(12,5))
ax = sns.countplot('Skill Moves', data = players,
                   palette= palette)
plt.ylabel('Total')
plt.show()
```



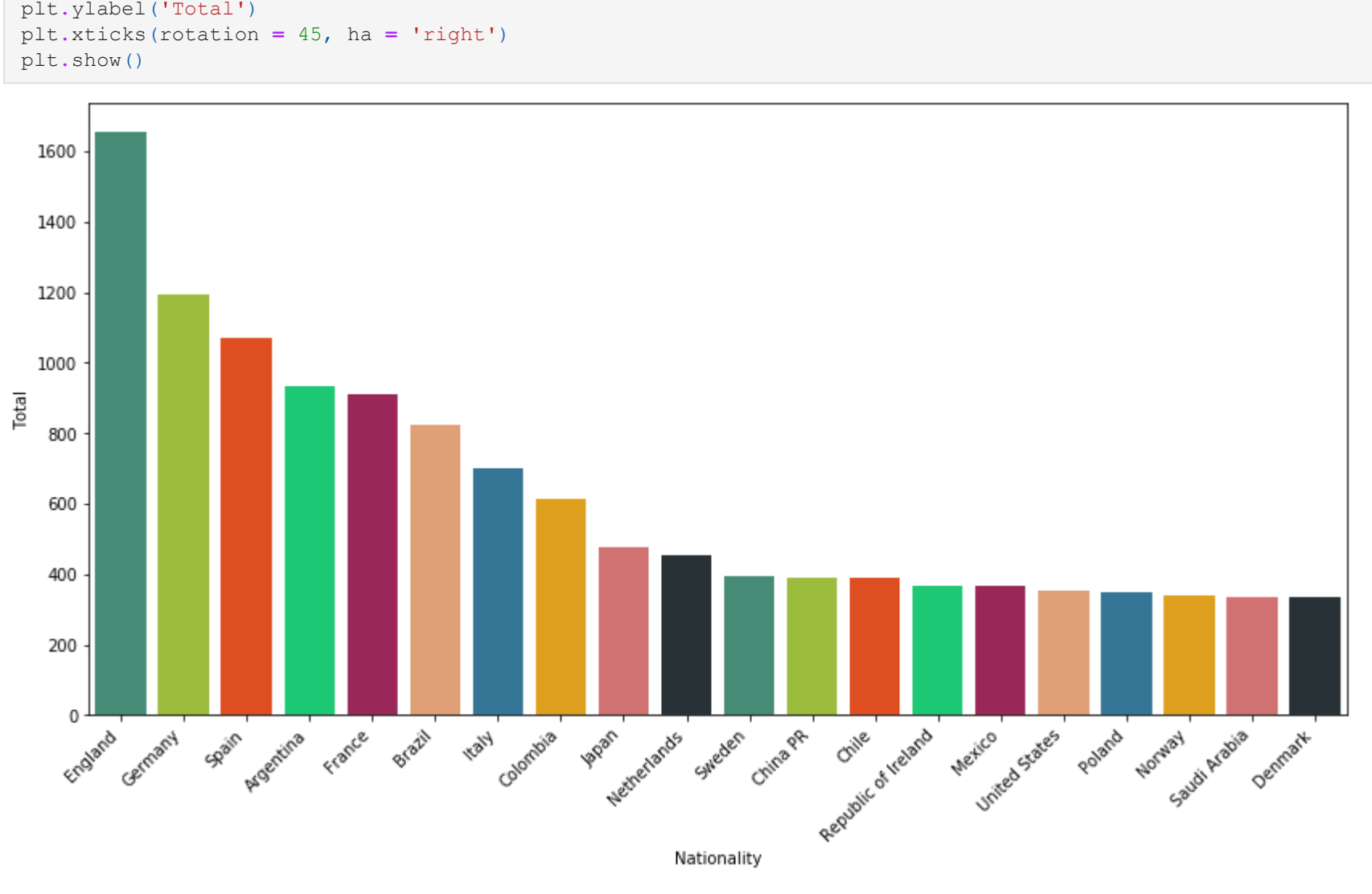
In [19]:
```
plt.figure(figsize=(12,5))
plt.xticks(rotation = 45, ha = 'right')
ax = sns.countplot('Work Rate', data = players,
                   palette= palette)
plt.ylabel('Total')
plt.show()
```



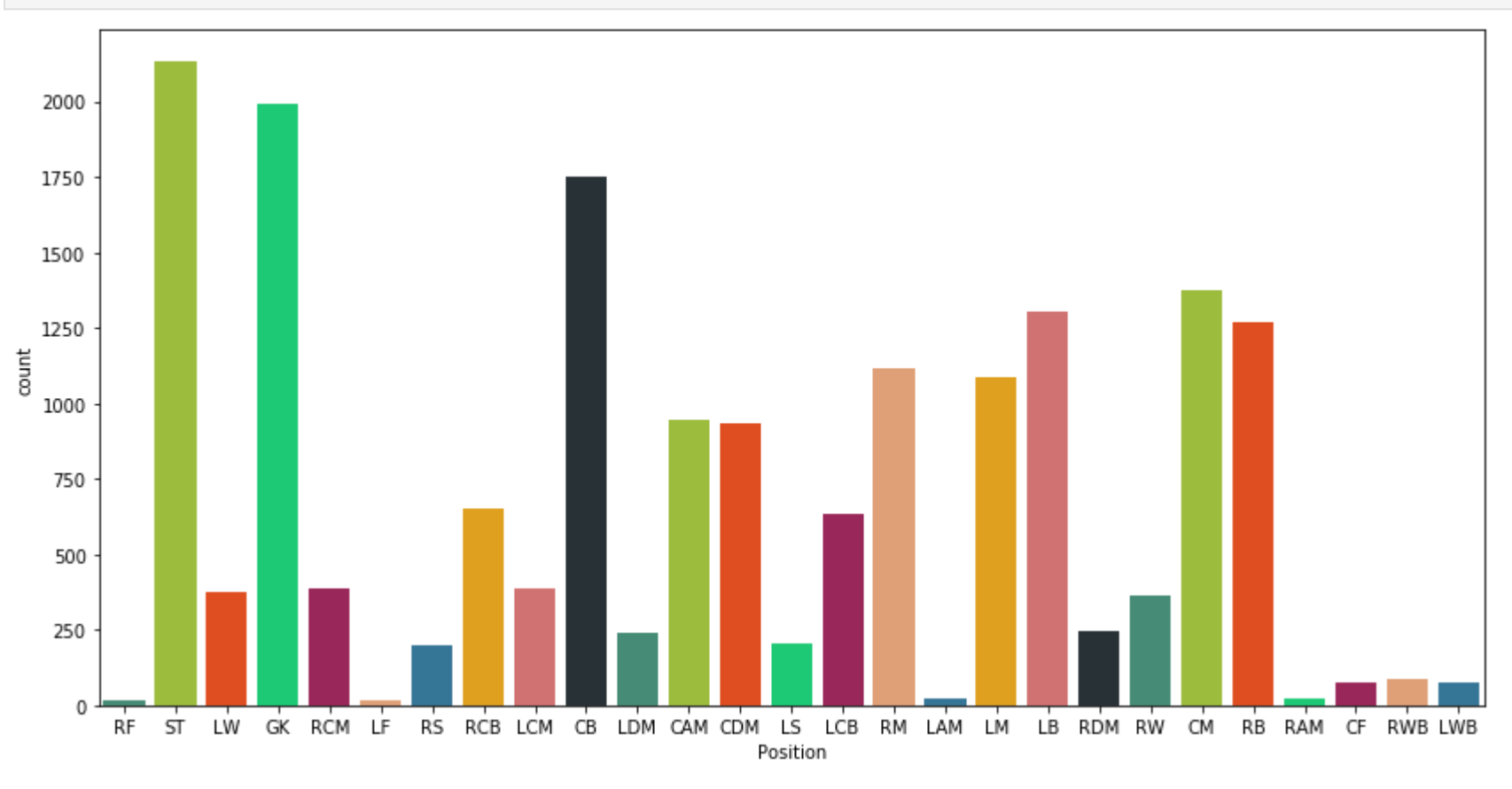
In [20]:
```
plt.figure(figsize=(12,5))
ax = sns.countplot('Age', data = players,
                    palette= palette)
plt.ylabel('Total')
plt.xticks(rotation = 45, ha = 'right', size=16)
plt.show()
```



In [21]:
```
plt.figure(figsize = (14,7))
ax = sns.countplot('Nationality', data = players ,
                    order = players['Nationality'].value_counts()[:20].index,
                    palette= palette)
plt.ylabel('Total')
plt.xticks(rotation = 45, ha = 'right')
plt.show()
```



In [22]:
```
plt.figure(figsize = (14,7))
sns.countplot(x = 'Position', data = players, palette = palette);
```



In [23]:
```
#### With these functions I will create new data, the data selection is based on the grouped
#### statistics that shows the fifa

def defending(players):
    return int(round((players[['Interceptions','Marking', 'StandingTackle',
                               'SlidingTackle', 'HeadingAccuracy']].mean()).mean()))

def passing(players):
    return int(round((players[['Vision', 'Crossing', 'ShortPassing',
                               'LongPassing','FKAccuracy','Curve']].mean()).mean()))

def dribbling(players):
    return int(round((players[['Agility', 'Balance', 'Reactions',
                               'BallControl', 'Dribbling']].mean()).mean()))

def shooting(players):
    return int(round((players[['Positioning', 'Finishing', 'ShotPower',
                               'LongShots', 'Volleys', 'Penalties']].mean()).mean()))

def pace(players):
    return int(round((players[['Acceleration', 'SprintSpeed']].mean()).mean()))

def physical(players):
    return int(round((players[['Jumping', 'Stamina', 'Strength', 'Aggression',
                               'Composure']].mean()).mean()))

#### This function is different, here I create a fact that is not in any statesman.
#### I take 50 points to the player's favorite foot and then I add the total of stars to
#### 10 that he has on his bad foot, so a player who has the right foot preferred and has
#### a star on his bad foot has 60 ambidextrous points (50 + 10) and one that has 5 stars
#### has 100 (50 + 50) in this way I can get what is the player's ambidextrous percentage.
#### If I have two players with the same statistics with this one I can differentiate them
#### because if they are the same I can choose with the one that best hit them with both feet.

def ambidextrous_func(players):
    return (int(players) * 10) + 50
```

In [24]:
```
#### Start a counter to see how long it took to execute all functions

timeFinish = 0
start_time = time()
```

In [25]:
```
players['Defending'] = players.apply(defending, axis=1)
players['Passing'] = players.apply(passing, axis=1)
players['Dribbling'] = players.apply(dribbling, axis=1)
players['Shooting'] = players.apply(shooting, axis=1)
players['Pace'] = players.apply(pace, axis=1)
players['Physical'] = players.apply(physical, axis=1)
players['Ambidextrous'] = players['Weak Foot'].apply(ambidextrous_func)
timeFinish += (time() - start_time)
print('Ending - time: ' + str(timedelta(seconds=timeFinish)))
```

```
Ending - time: 0:07:42.775567
```

In [26]:
```
len(players.columns)
```

Out[26]: 96

In [27]:
```
#### The dataset has 96 columns, let's put together another one with the ones that are going
#### to be used

data = players[['ID','Name','Defending','Passing','Dribbling','Shooting',
                'Pace','Physical','Ambidextrous','GKDiving',
                'GKHandling', 'GKKicking', 'GKReflexes','Flag','Age',
                'Nationality', 'Photo', 'Club_Logo', 'Club', 'Position',
                'Value', 'Wage','Preferred_Foot','Body_Type','Jersey_Number',
                'Joined','Loaned_From','Height','Weight','Contract_Valid',
                'Overall','Potential']]
```

In [28]:
```
data.head(1)
```

Out[28]:

	ID	Name	Defending	Passing	Dribbling	Shooting	Pace	Physical	Ambidextrous	GKDiving	...	Preferred_Foot	Body_Type	Jersey_N
0	158023	L. Messi	36	90	95	88	88	69	90	6.0	...	Left	Normal	

1 rows × 32 columns

In [29]:
```
plt.figure(figsize = (14,10))
sns.heatmap(players[['Defending', 'Passing', 'Dribbling', 'Shooting', 'Pace',
       'Physical', 'Ambidextrous', 'Overall', 'Potential','GKDiving', 'GKHandling',
        'GKKicking','GKReflexes']].corr(), annot = True,
                        linewidths=.5, cmap= 'Reds')
plt.show()
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: