In [3]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
sns.set(style="whitegrid")
import matplotlib.pyplot as plt
from collections import Counter
%matplotlib inline
```

In [4]:
```
# ignore warnings
import warnings
warnings.filterwarnings('ignore')
```

In [5]:
```
fifa = pd.read_csv(r'C:\Users\Me\OneDrive\Data Science\0504\5th - Seaborn, Eda pract
```

In [6]:
```
fifa.head() #Preview the dataset
```

Out[6]:

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

5 rows × 89 columns

In [7]:
```
fifa.info() #View summary of dataset
```

```
<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
 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
```

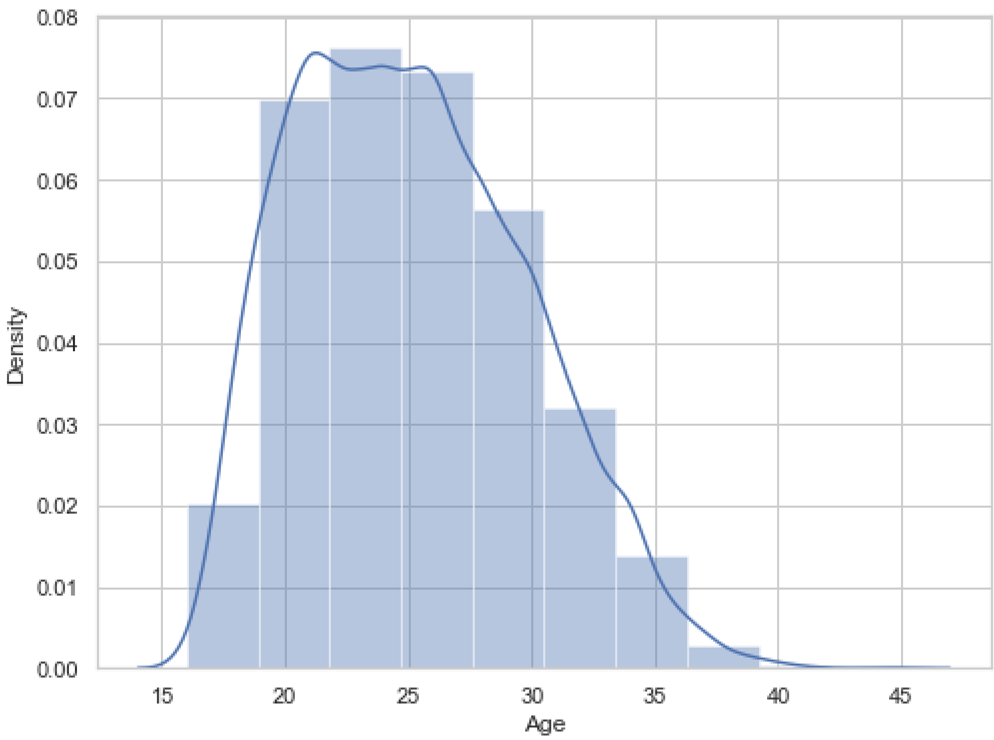
In [8]:
```
fifa['Body Type'].value_counts()
```

Out[8]:
```
Normal                 10595
Lean                    6417
Stocky                  1140
PLAYER_BODY_TYPE_25        1
Shaqiri                    1
Akinfenwa                  1
Neymar                     1
Messi                      1
Courtois                   1
C. Ronaldo                 1
Name: Body Type, dtype: int64
```

Comment

- This dataset contains 89 variables.
- Out of the 89 variables, 44 are numerical variables. 38 are of float64 data type and remaining 6 are of int64 data type.
- The remaining 45 variables are of character data type.

In [9]:
```
f, ax = plt.subplots(figsize=(8,6))
x = fifa['Age']
ax = sns.distplot(x, bins=10)
plt.show()
```

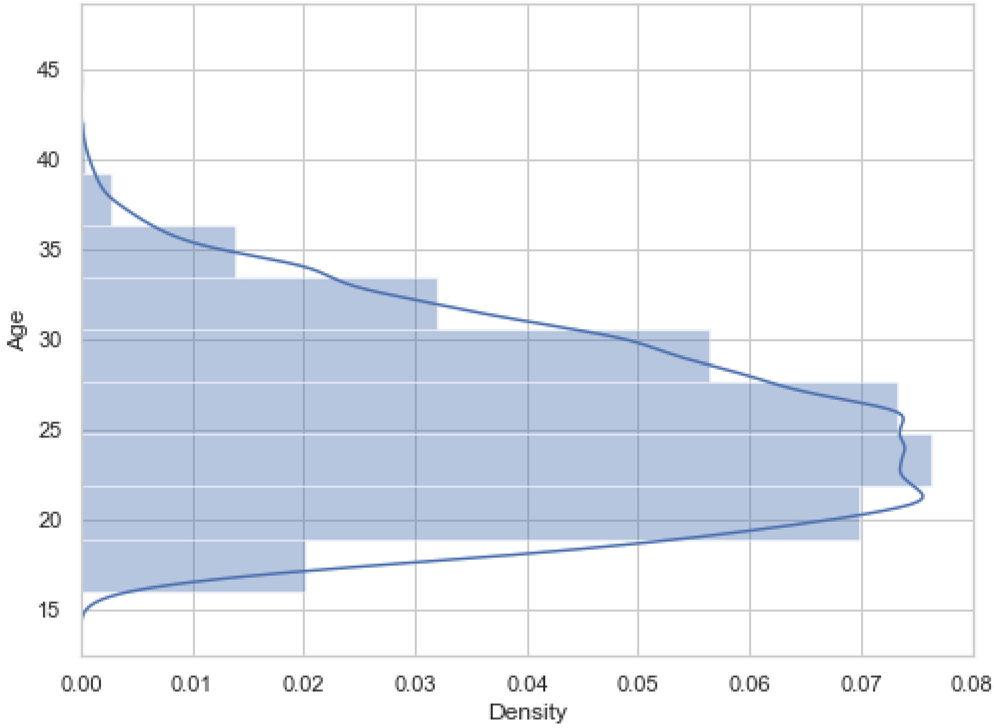


Histogram: The bars represent a histogram that bins the 'Age' data into discrete intervals. The height of each bar shows the relative number of players within each age bin. In this plot, the bins appear to be of equal width, and the distribution of player ages looks roughly normal but is slightly skewed to the right, with a peak somewhere around the mid-20s.

Kernel Density Estimate (KDE): The smooth line overlaying the histogram is the KDE, which provides a continuous probability density curve of the 'Age' variable. This curve is useful for seeing the shape of the distribution, and in this case, it highlights that most players fall into the younger age groups with a gradual decrease in density as age increases.

This graph illustrates the age distribution of players in the FIFA video game. The majority of players are concentrated in their early to mid-20s, which is the peak performance age range for professional athletes in this dataset. The density of players gradually decreases with age, which suggests there are fewer older players, reflecting perhaps both a natural career progression and a retirement age for professional footballers. Notably, there are very few players below the age of 20 or above 40, which are the tails of the distribution. The slight right skewed indicates that while there are fewer older players, the decline is not as rapid as the incline during the young adult years

In [10]:
```
f, ax = plt.subplots(figsize=(8,6))
x = fifa['Age']
ax = sns.distplot(x, bins=10, vertical = True)
plt.show()
```

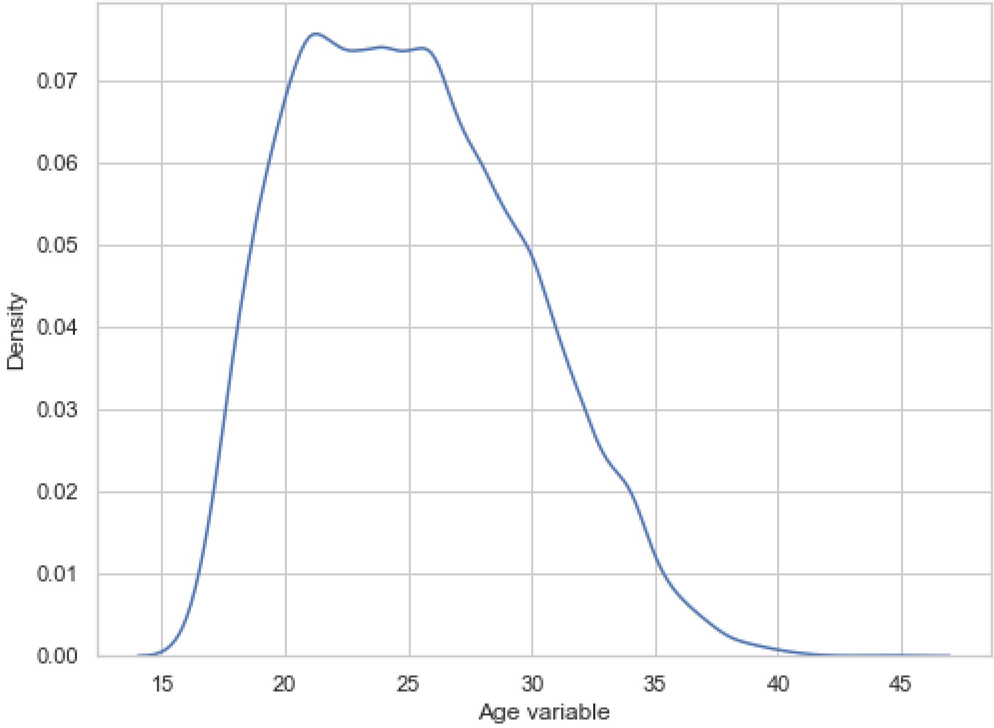


Seaborn Kernel Density Estimation (KDE) Plot

- The `kernel density estimate (KDE)` plot is a useful tool for plotting the shape of a distribution.
- Seaborn kdeplot is another seaborn plotting function that fits and plot a univariate or bivariate kernel density estimate.
- Like the histogram, the KDE plots encode the density of observations on one axis with height along the other axis.
- We can plot a KDE plot as follows-

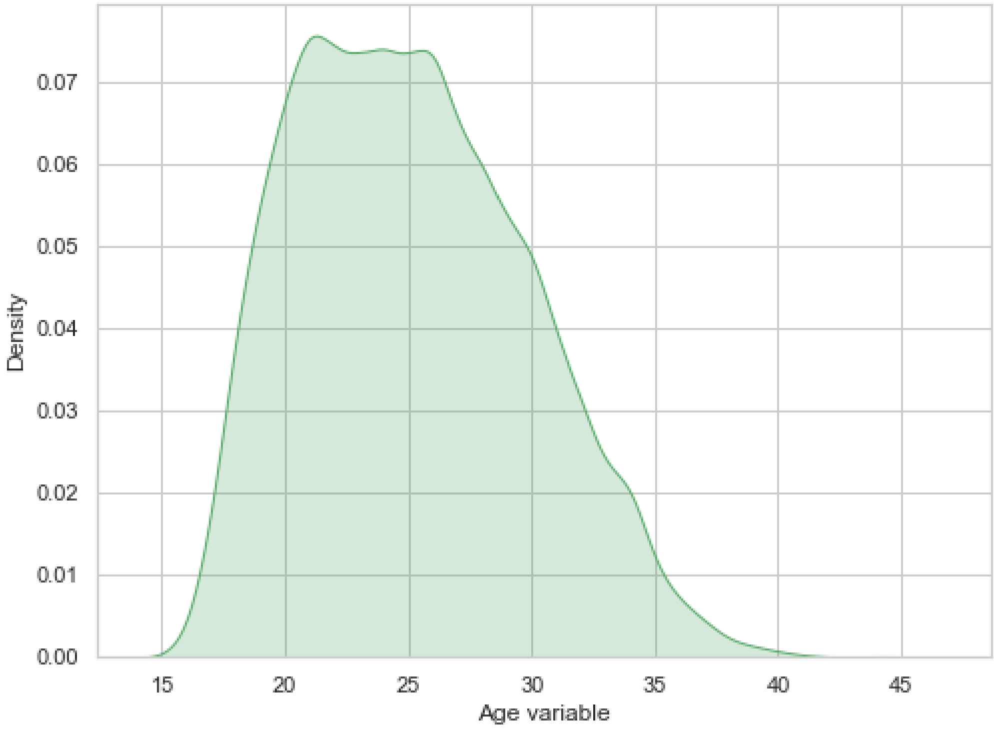
In [11]:
```
f, ax = plt.subplots(figsize=(8,6))
x = fifa['Age']
x = pd.Series(x, name="Age variable")
```

```
ax = sns.kdeplot(x)
plt.show()
```

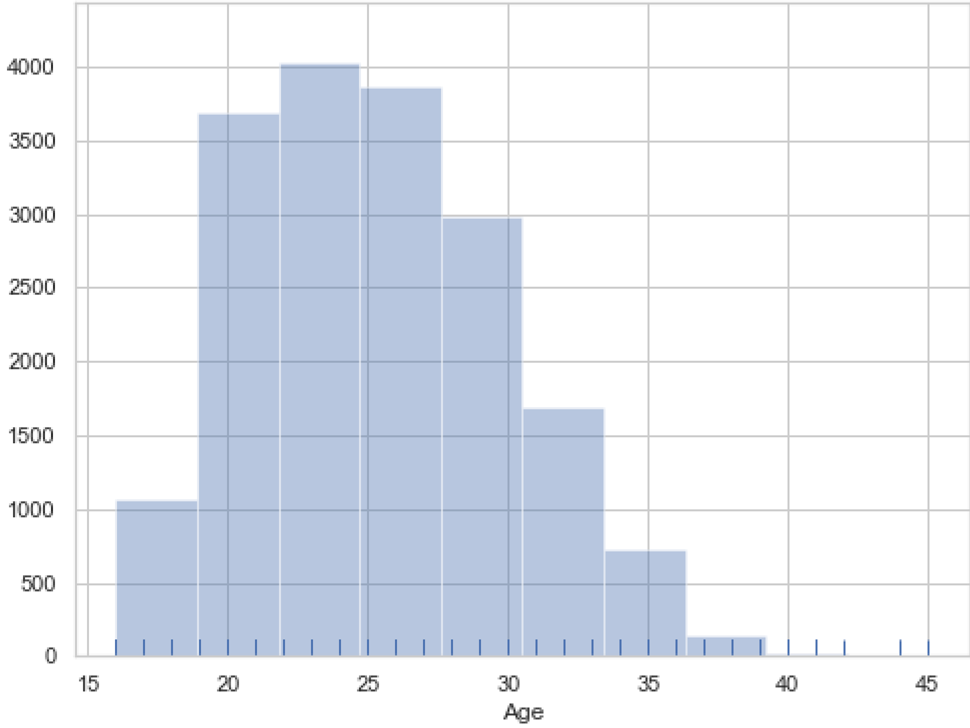


shading under the density curve

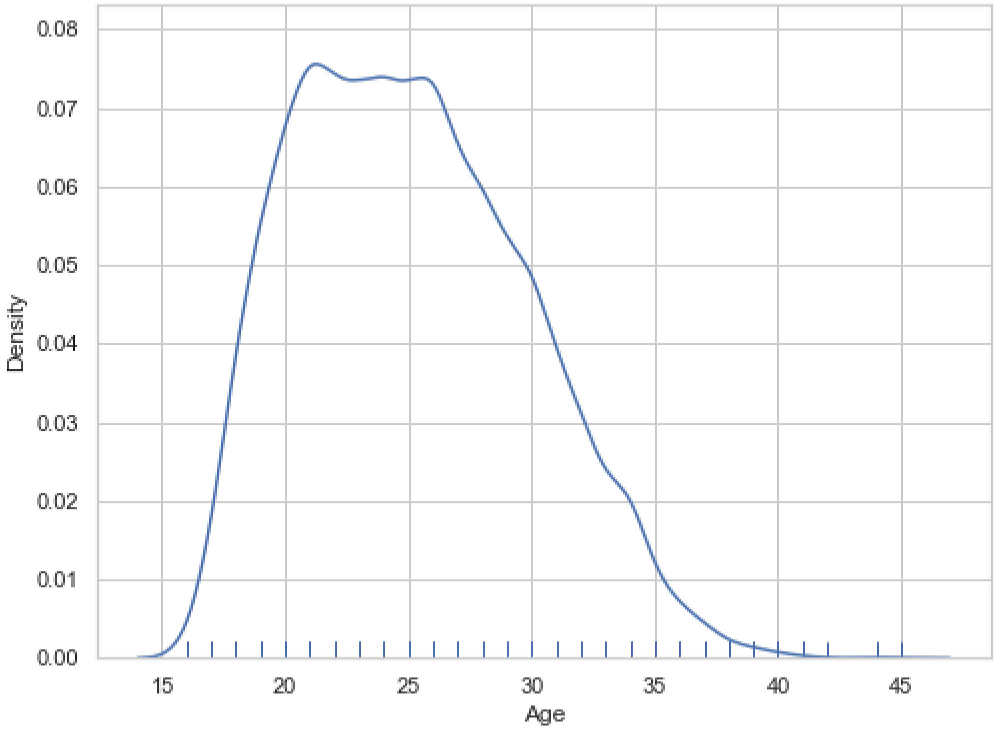
In [12]:
```
f, ax = plt.subplots(figsize=(8,6))
x = fifa['Age']
x = pd.Series(x, name="Age variable")
ax = sns.kdeplot(x, shade=True, color='g')
plt.show()
```



In [13]:
```
f, ax = plt.subplots(figsize=(8,6))
x = fifa['Age']
ax = sns.distplot(x, kde=False, rug=True, bins=10)
plt.show()
```



In [14]:
```
f, ax = plt.subplots(figsize=(8,6))
x = fifa['Age']
ax = sns.distplot(x, hist=False, rug=True, bins=10)
plt.show()
```



fifa19['Preferred Foot'].nunique() #Check number of unique values in Preferred Foot variable

In [15]:
```
fifa['Preferred Foot'].nunique()
```

Out[15]: 2

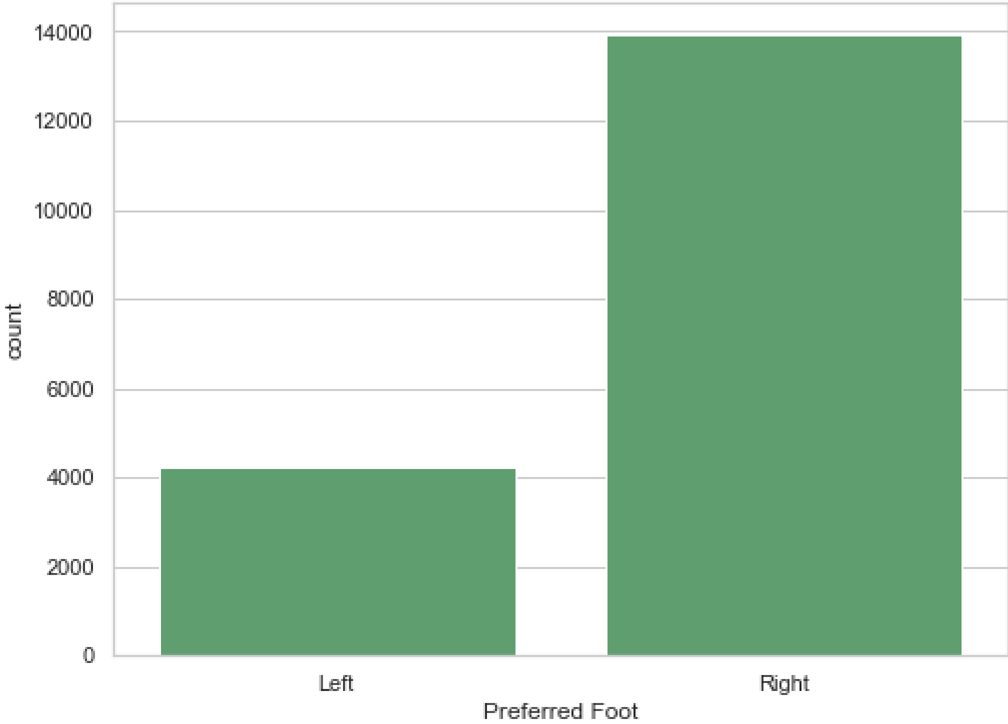
two types of unique values in Preferred Foot variable.

In [17]:
```
fifa['Preferred Foot'].value_counts() #Check frequency distribution of values in Pre
```

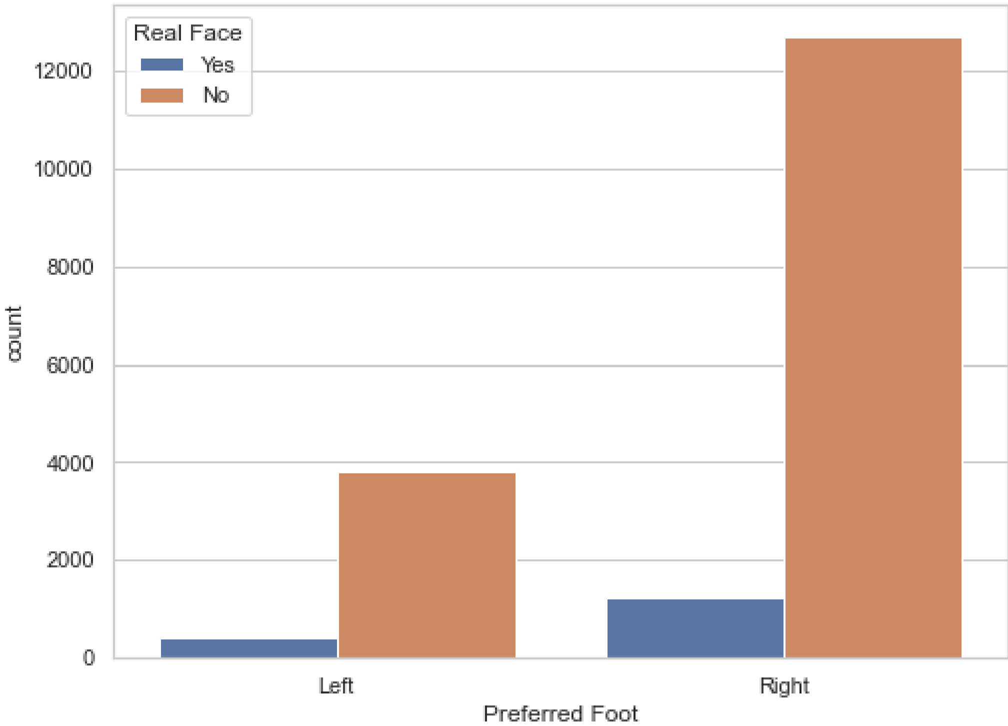
Out[17]:
```
Right    13948
Left      4211
Name: Preferred Foot, dtype: int64
```

The Preferred Foot variable contains two types of values - Right and Left

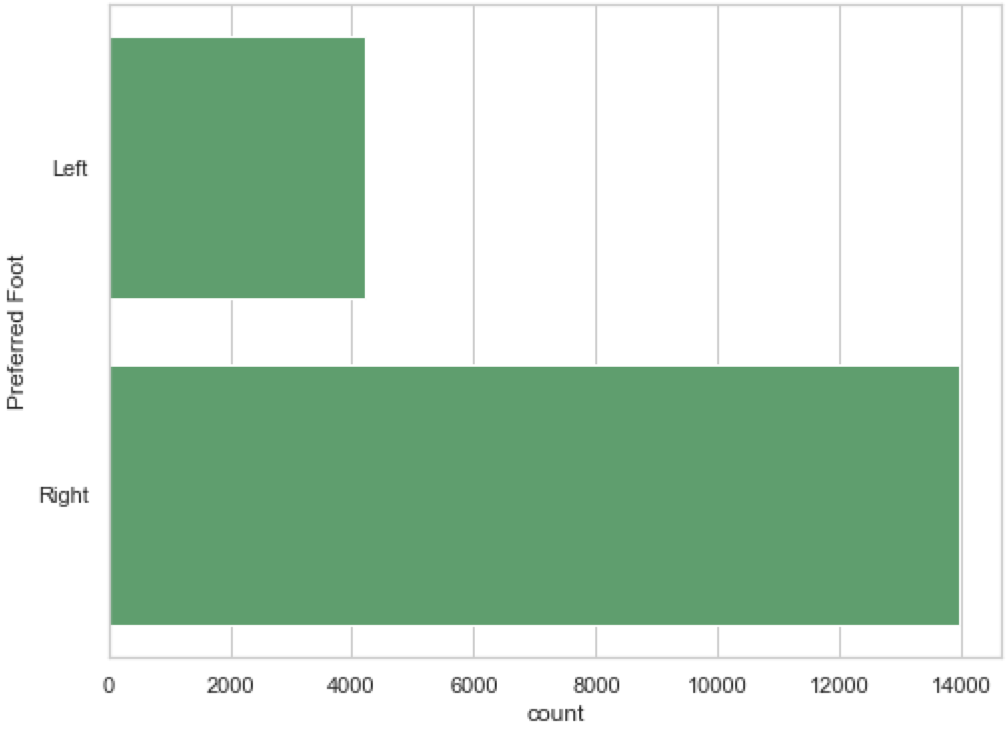
In [24]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.countplot(x="Preferred Foot", data=fifa, color="g")
plt.show()
```



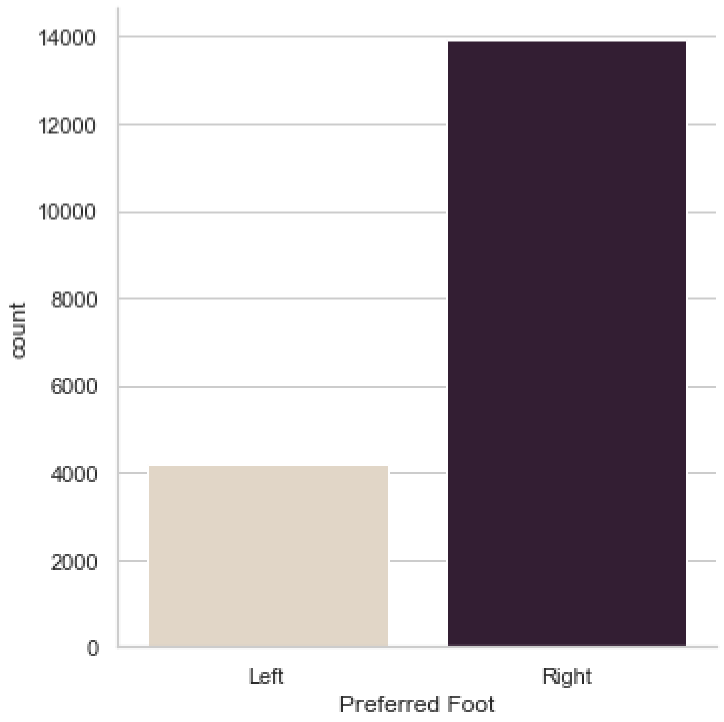
In [25]:
```
#value counts for two categorical variables
f, ax = plt.subplots(figsize=(8, 6))
sns.countplot(x="Preferred Foot", hue="Real Face", data=fifa)
plt.show()
```



In [27]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.countplot(y="Preferred Foot", data=fifa, color="g")
plt.show()
```



In [29]:
```
g = sns.catplot(x="Preferred Foot", kind="count", palette="ch:.25", data=fifa)
#Seaborn catplot() function to draw a countplot()
```



In [30]:
```
fifa['International Reputation'].nunique() #Check the number of unique values in Int
```

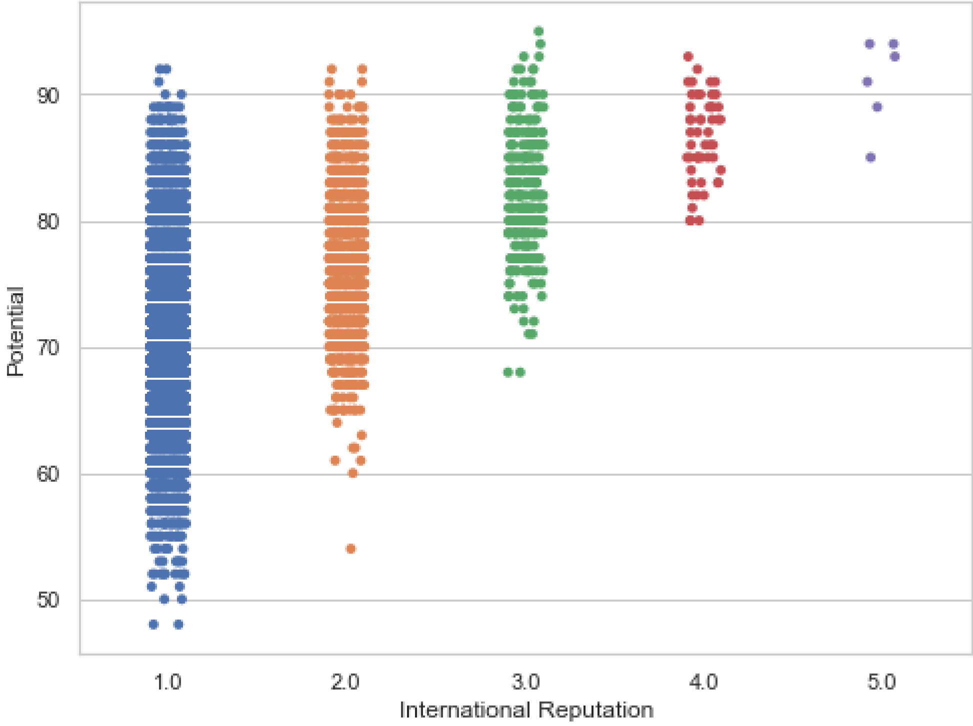
Out[30]: 5

In [32]:
```
fifa['International Reputation'].value_counts()  #Check the distribution of values i
```

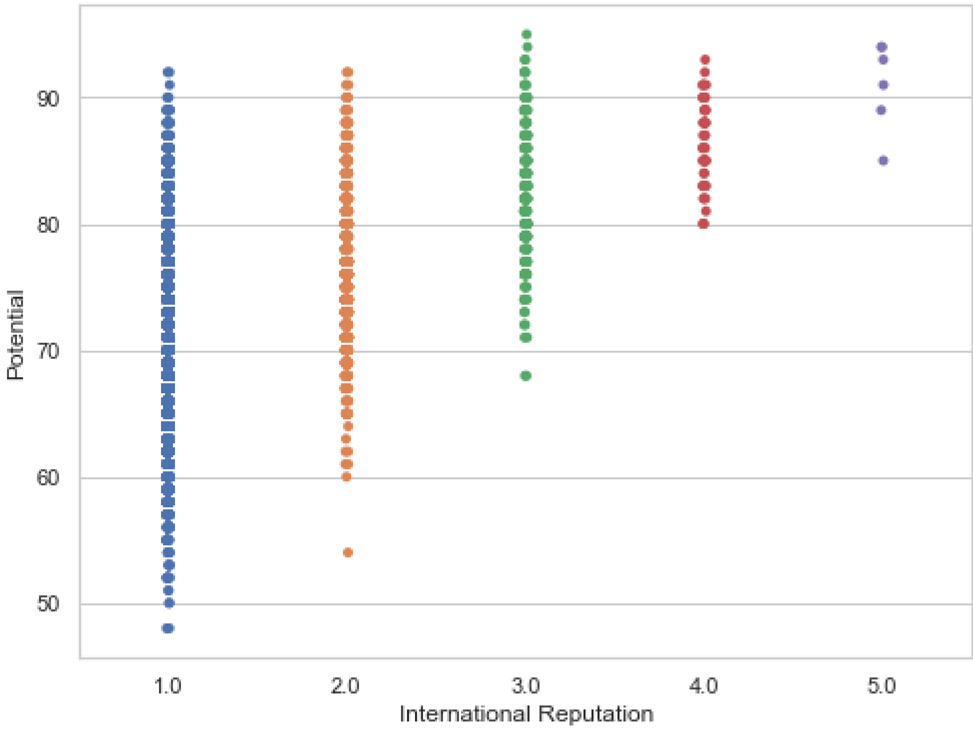
Out[32]:
```
1.0    16532
2.0     1261
3.0      309
```

```
4.0        51
5.0         6
Name: International Reputation, dtype: int64
```

In [34]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.stripplot(x="International Reputation", y="Potential", data=fifa)
plt.show()
#plot a stripplot with International Reputation as categorical variable and Potentia
```

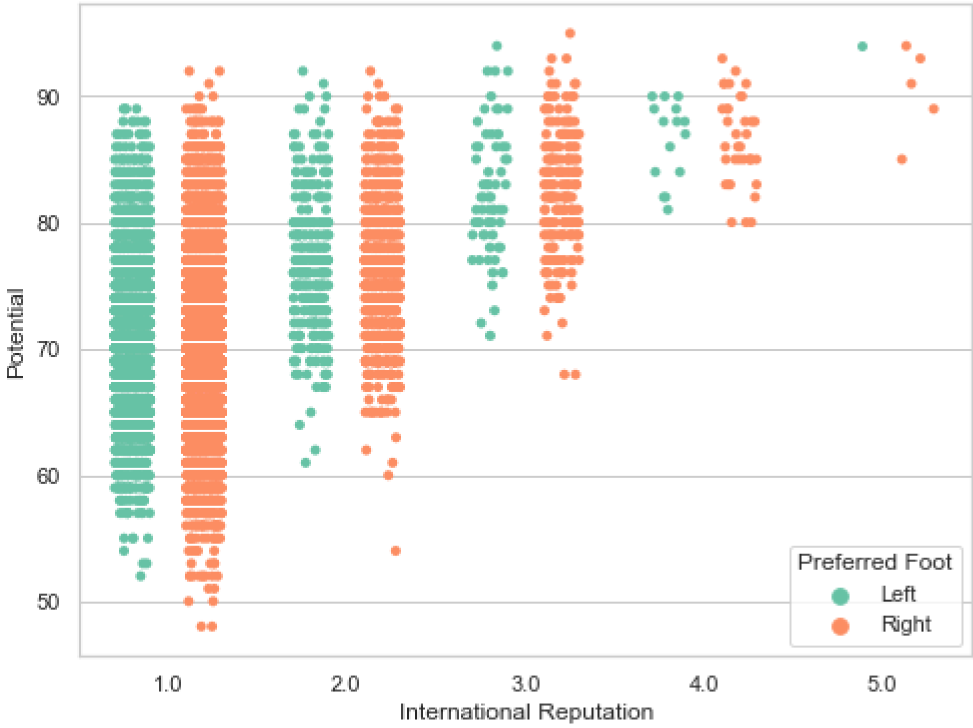


In [35]:
```
#add jitter to bring out the distribution of values
f, ax = plt.subplots(figsize=(8, 6))
sns.stripplot(x="International Reputation", y="Potential", data=fifa, jitter=0.01)
plt.show()
```



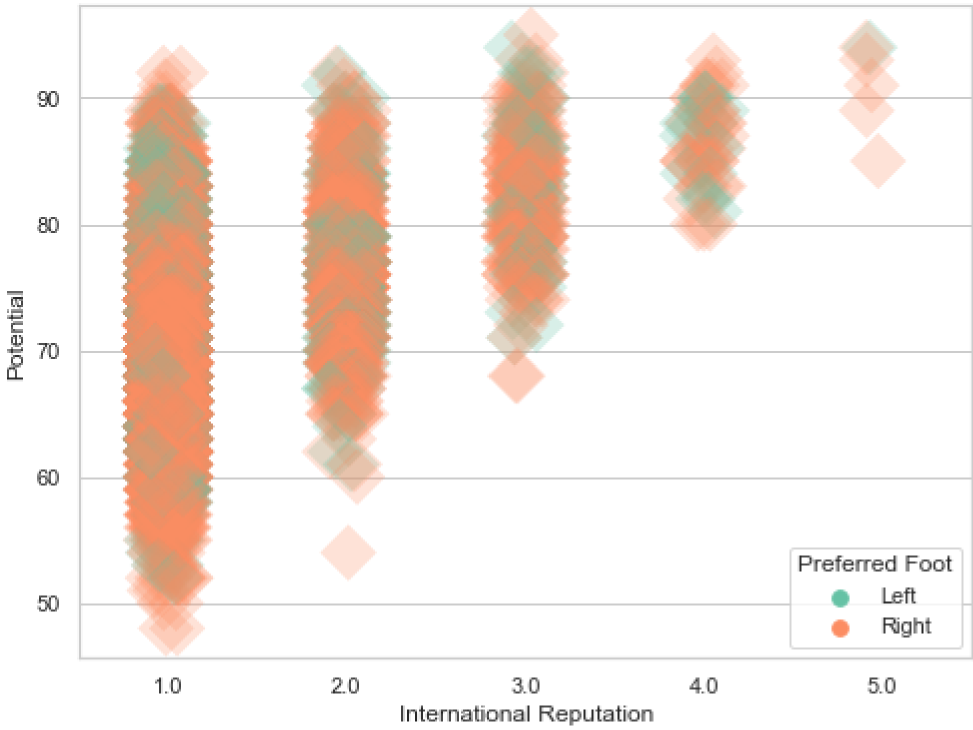
In [36]:
```
#nest the strips within a second categorical variable - Preferred Foot
f, ax = plt.subplots(figsize=(8, 6))
```

```
sns.stripplot(x="International Reputation", y="Potential", hue="Preferred Foot",
                   data=fifa, jitter=0.2, palette="Set2", dodge=True)
plt.show()
```



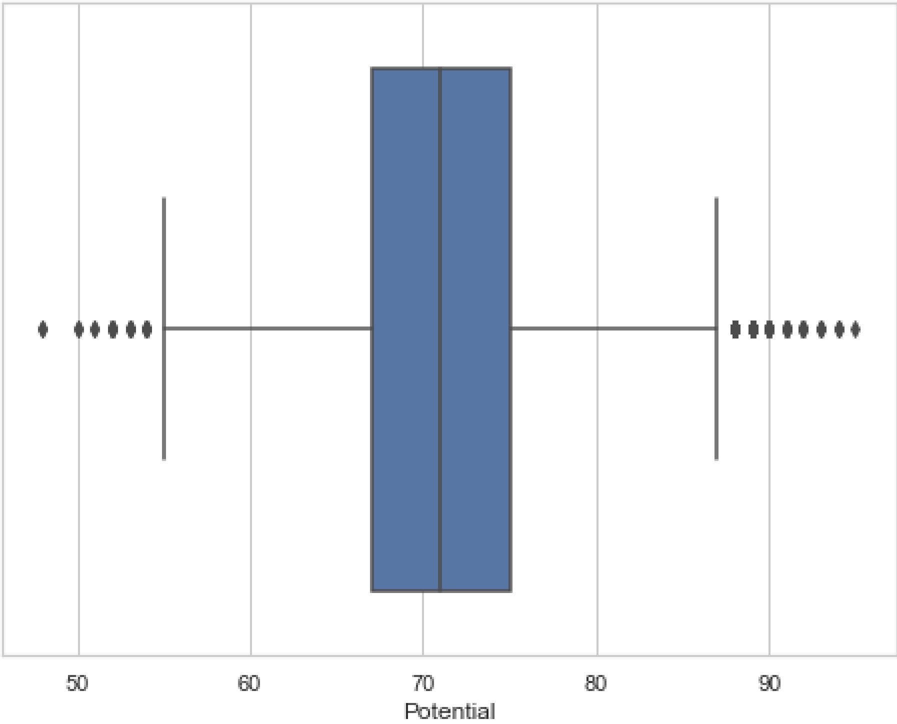
In [38]:

```
#strips with large points and different aesthetics
f, ax = plt.subplots(figsize=(8, 6))
sns.stripplot(x="International Reputation", y="Potential", hue="Preferred Foot",
                   data=fifa, palette="Set2", size=20, marker="D",
                   edgecolor="gray", alpha=.25)
plt.show()
```

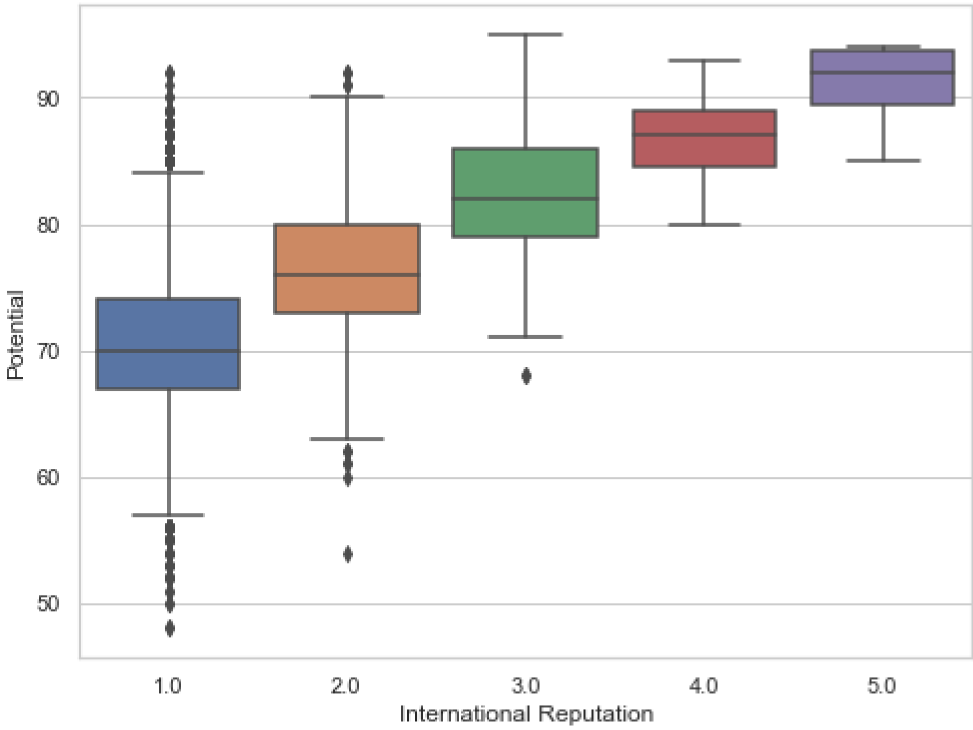


In [40]:

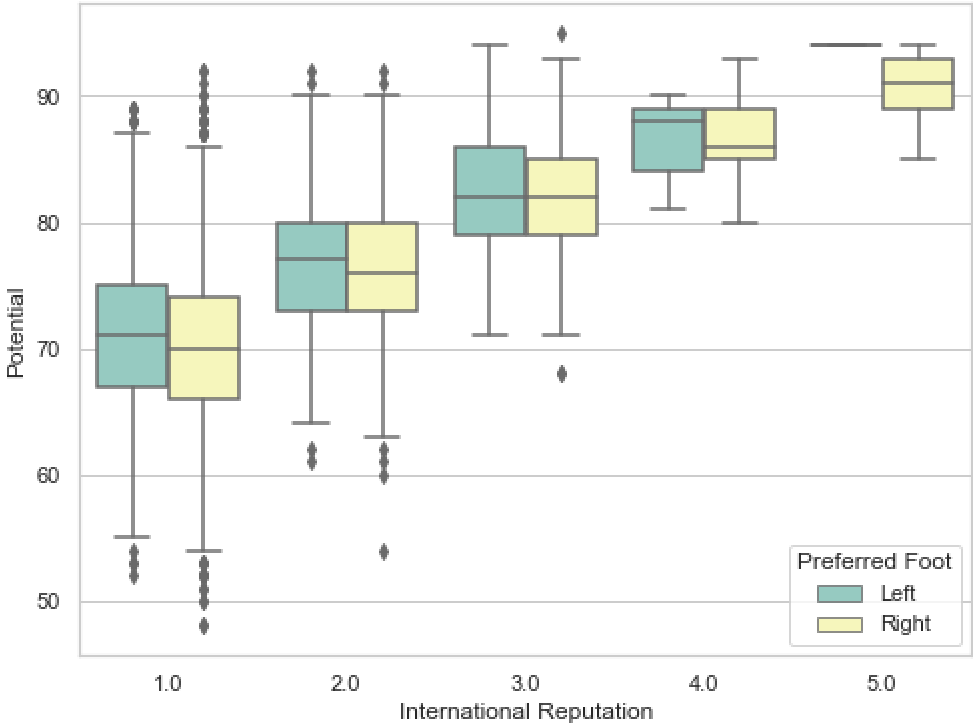
```
#boxplot of the Potential variable
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=fifa["Potential"])
plt.show()
```



In [43]:
```
#vertical boxplot grouped by the categorical variable International Reputation
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x="International Reputation", y="Potential", data=fifa)
plt.show()
```



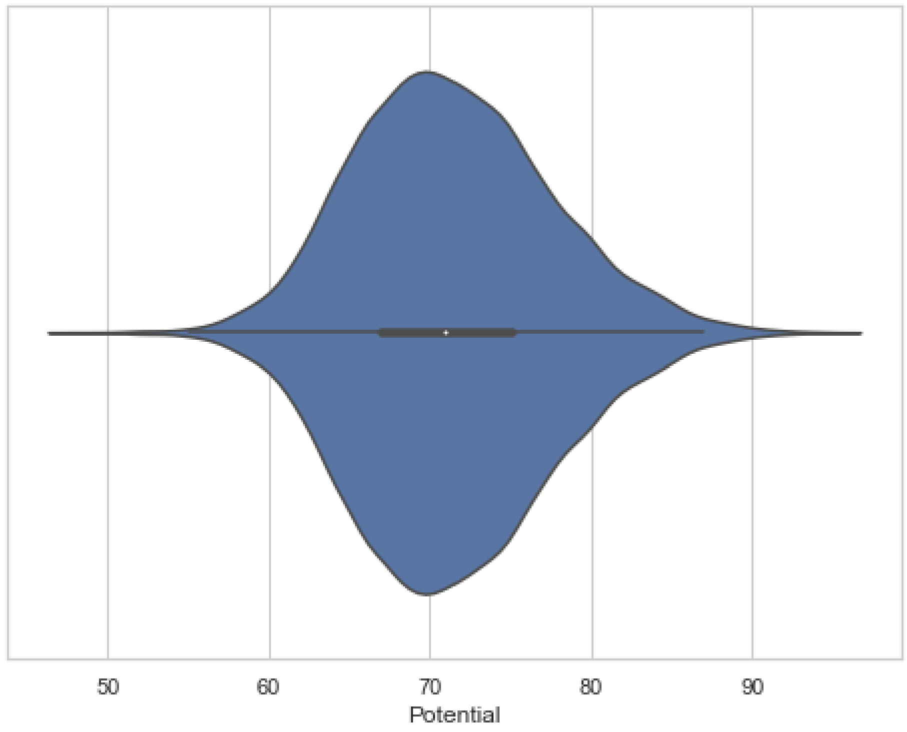
In [44]:
```
#boxplot with nested grouping by two categorical variables
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x="International Reputation", y="Potential", hue="Preferred Foot", data=
plt.show()
```



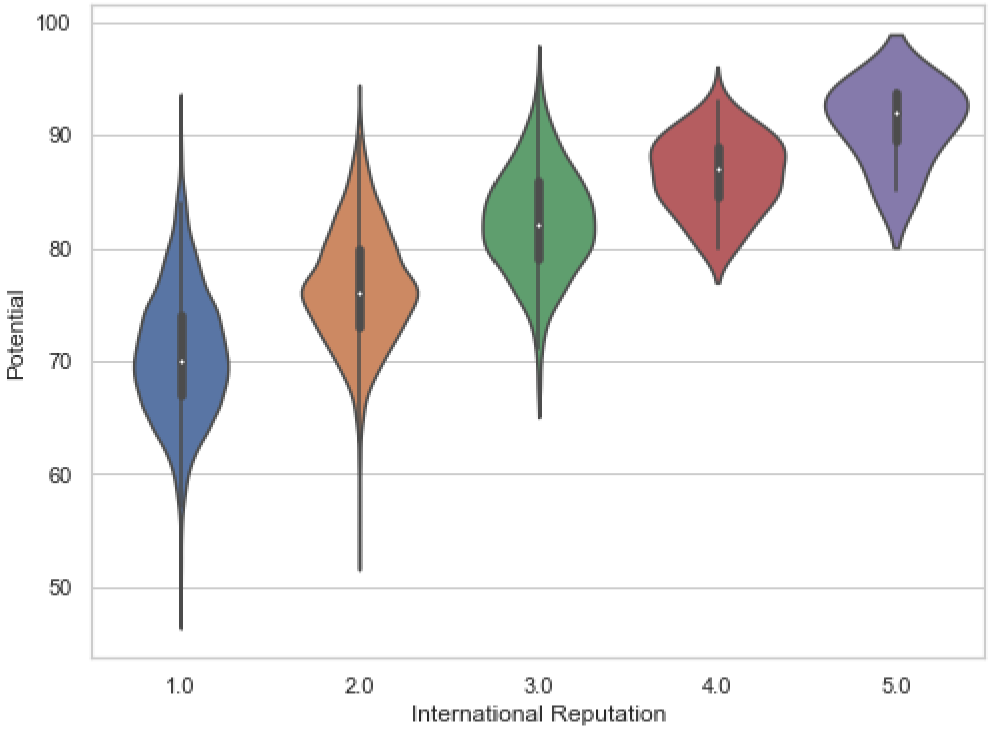
Seaborn `violinplot()` function

- This function draws a combination of boxplot and kernel density estimate.
- A violin plot plays a similar role as a box and whisker plot.
- It shows the distribution of quantitative data across several levels of one (or more) categorical variables such that those distributions can be compared.
- Unlike a box plot, in which all of the plot components correspond to actual datapoints, the violin plot features a kernel density estimation of the underlying distribution.

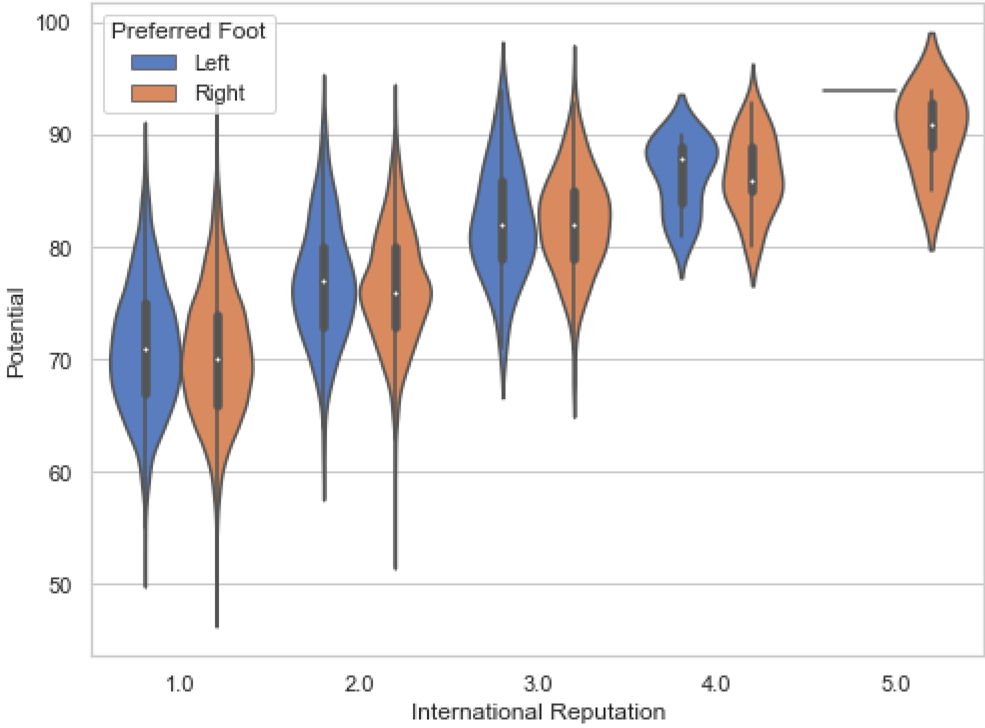
In [46]:
```
#violinplot of `Potential` variable
f, ax = plt.subplots(figsize=(8, 6))
sns.violinplot(x=fifa["Potential"])
plt.show()
```



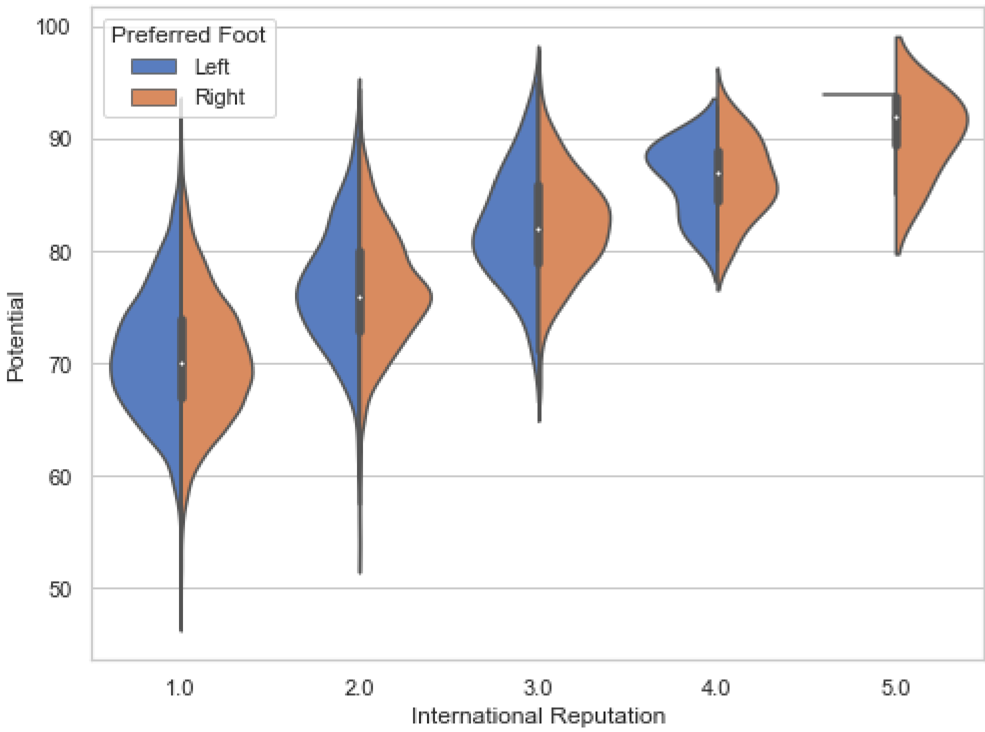
In [47]:
```
#vertical violinplot grouped by the categorical variable International Reputation
f, ax = plt.subplots(figsize=(8, 6))
sns.violinplot(x="International Reputation", y="Potential", data=fifa)
plt.show()
```



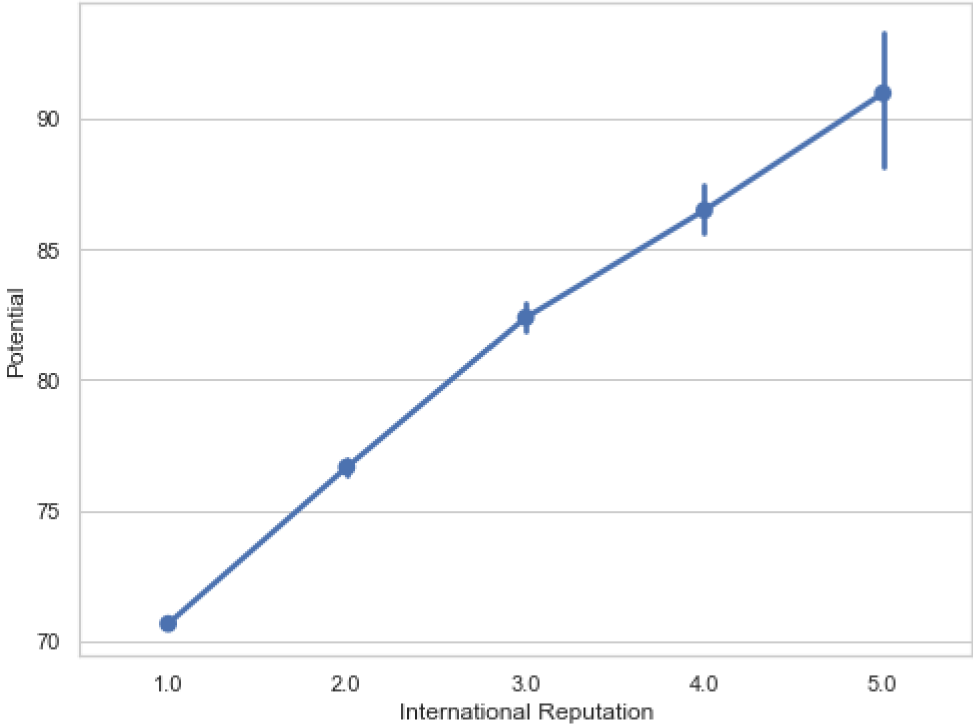
In [48]:
```
#violinplot with nested grouping by two categorical variables
f, ax = plt.subplots(figsize=(8, 6))
sns.violinplot(x="International Reputation", y="Potential", hue="Preferred Foot", da
plt.show()
```



In [49]:
```
#split violins to compare the across the hue variable
f, ax = plt.subplots(figsize=(8, 6))
sns.violinplot(x="International Reputation", y="Potential", hue="Preferred Foot",
               data=fifa, palette="muted", split=True)
plt.show()
```

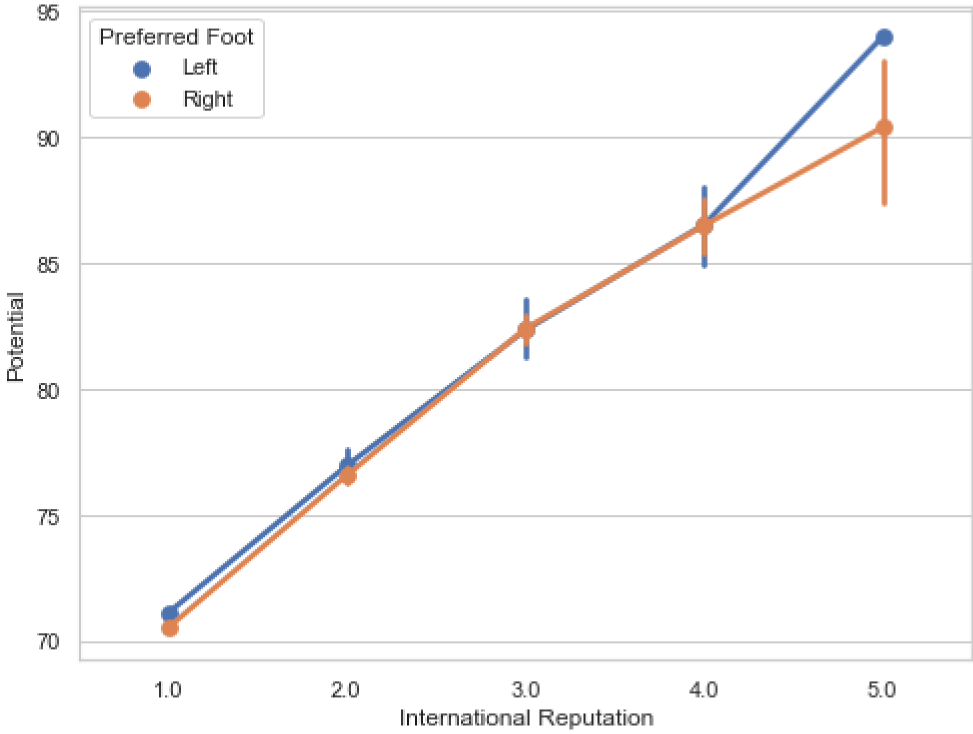


In [50]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.pointplot(x="International Reputation", y="Potential", data=fifa)
plt.show()
```



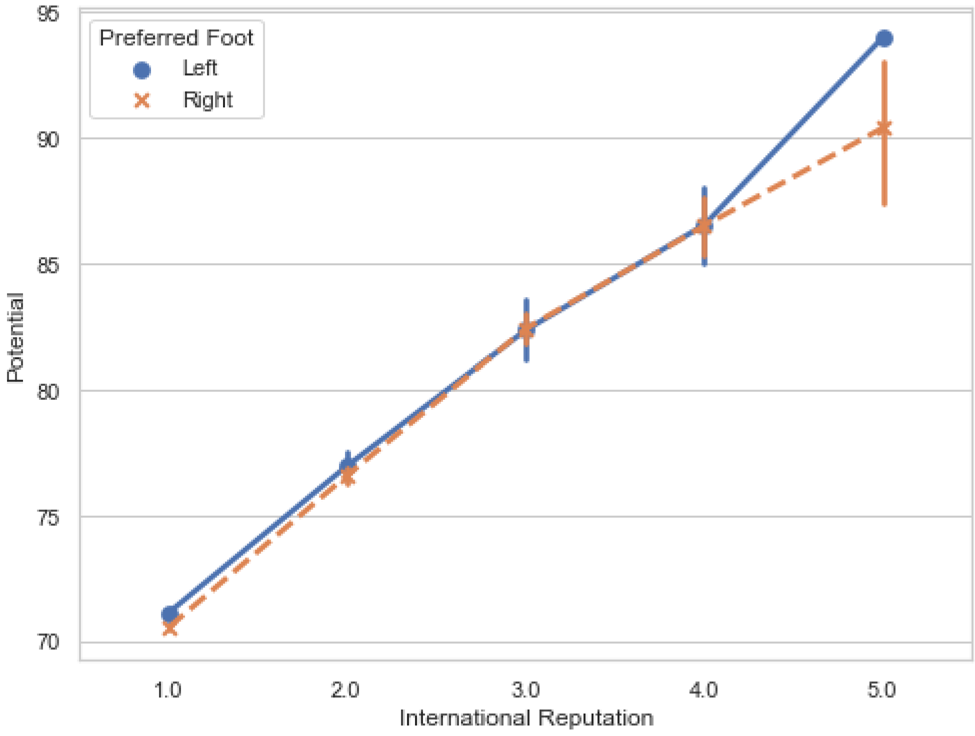
In [51]:

```
#vertical points with nested grouping by a two variables
f, ax = plt.subplots(figsize=(8, 6))
sns.pointplot(x="International Reputation", y="Potential", hue="Preferred Foot", dat
plt.show()
```

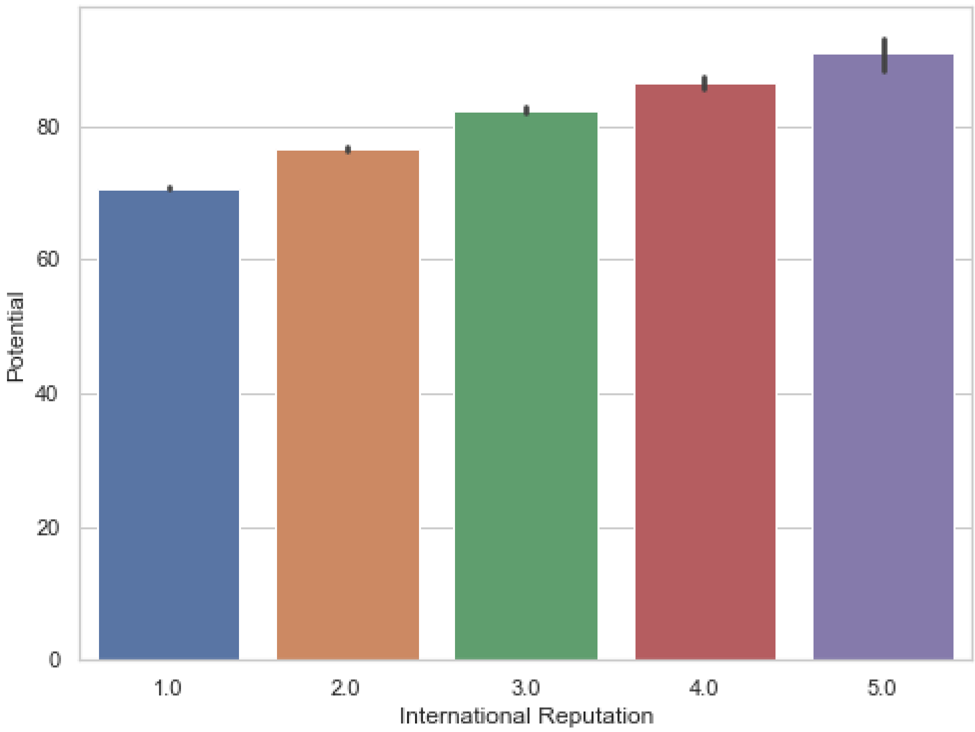


In [52]:

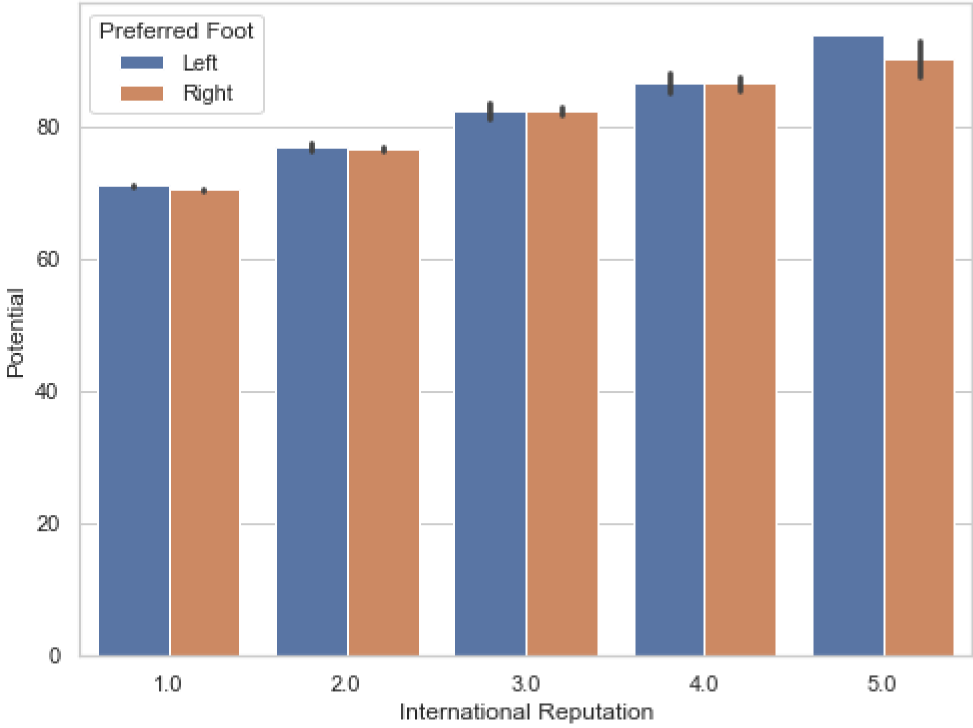
```
#different marker and line style for the hue levels
f, ax = plt.subplots(figsize=(8, 6))
sns.pointplot(x="International Reputation", y="Potential", hue="Preferred Foot",
              data=fifa, markers=["o", "x"], linestyles=["-", "--"])
plt.show()
```



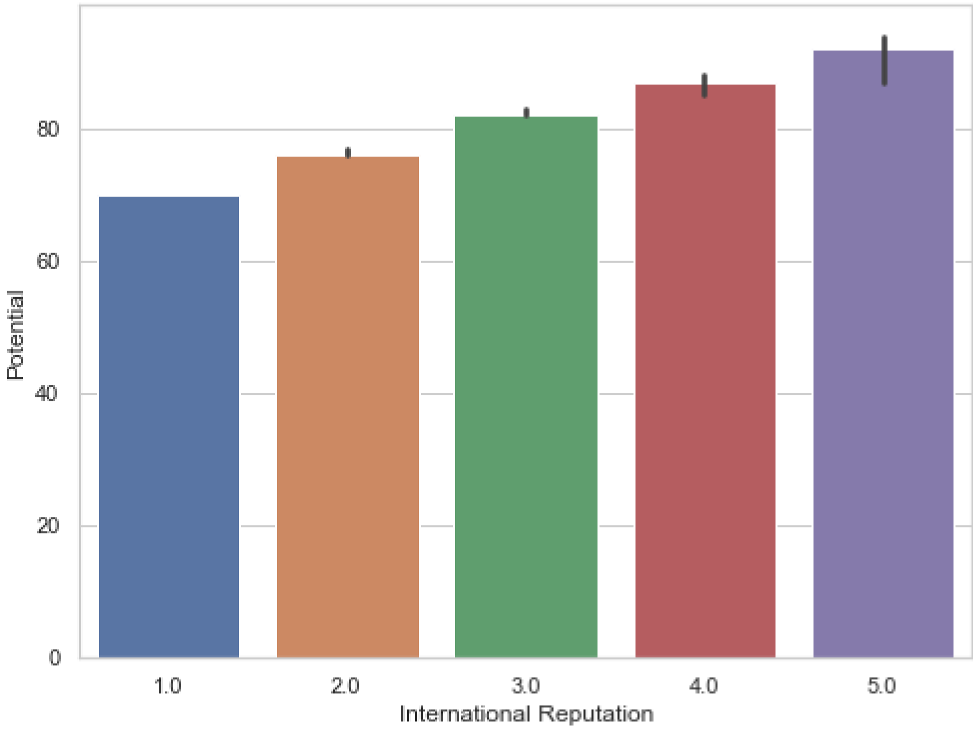
In [53]:
```
#plot a barplot
f, ax = plt.subplots(figsize=(8, 6))
sns.barplot(x="International Reputation", y="Potential", data=fifa)
plt.show()
```



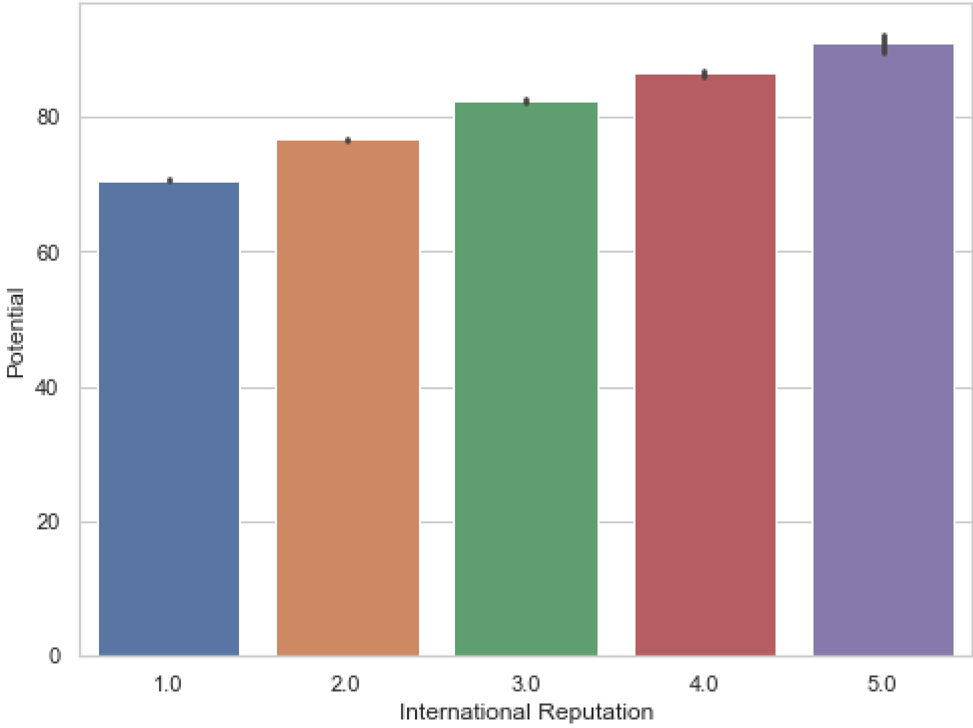
In [54]:
```
#vertical bars with nested grouping by a two variables
f, ax = plt.subplots(figsize=(8, 6))
sns.barplot(x="International Reputation", y="Potential", hue="Preferred Foot", data=
plt.show()
```



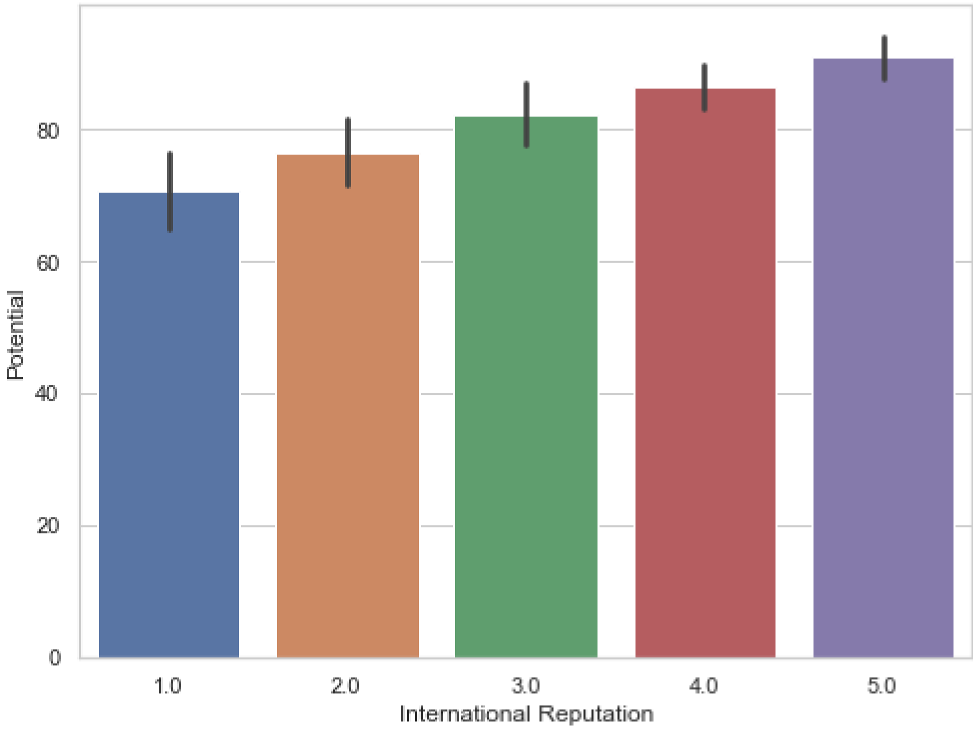
In [55]:
```
#using median as the estimate of central tendency
from numpy import median
f, ax = plt.subplots(figsize=(8, 6))
sns.barplot(x="International Reputation", y="Potential", data=fifa, estimator=median
plt.show()
```



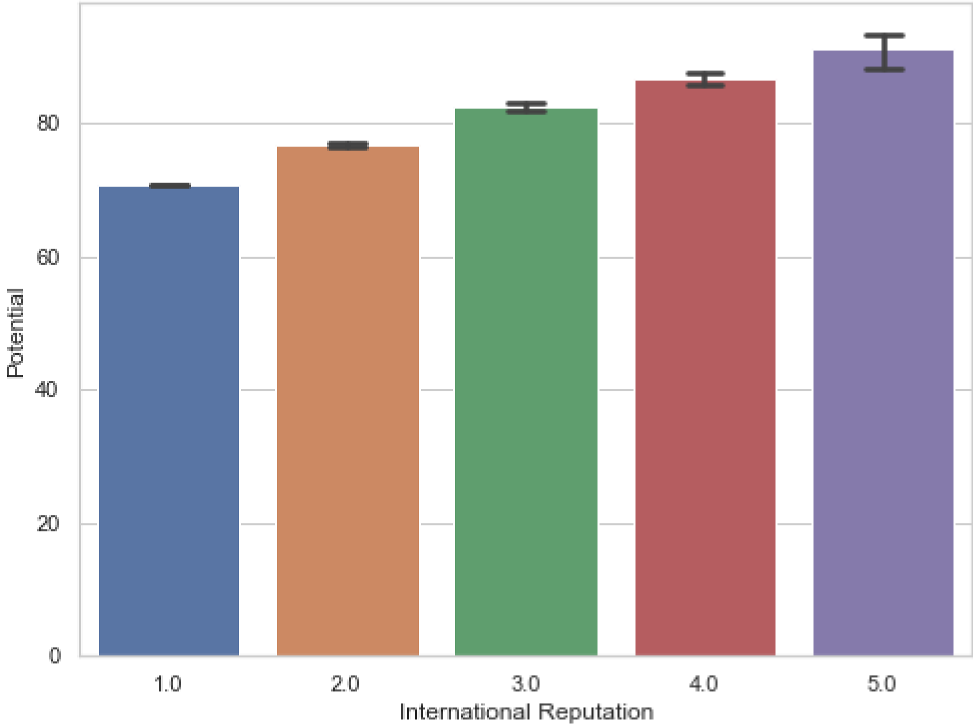
In [56]:
```
#showing the standard error of the mean with the error bars
f, ax = plt.subplots(figsize=(8, 6))
sns.barplot(x="International Reputation", y="Potential", data=fifa, ci=68)
plt.show()
```



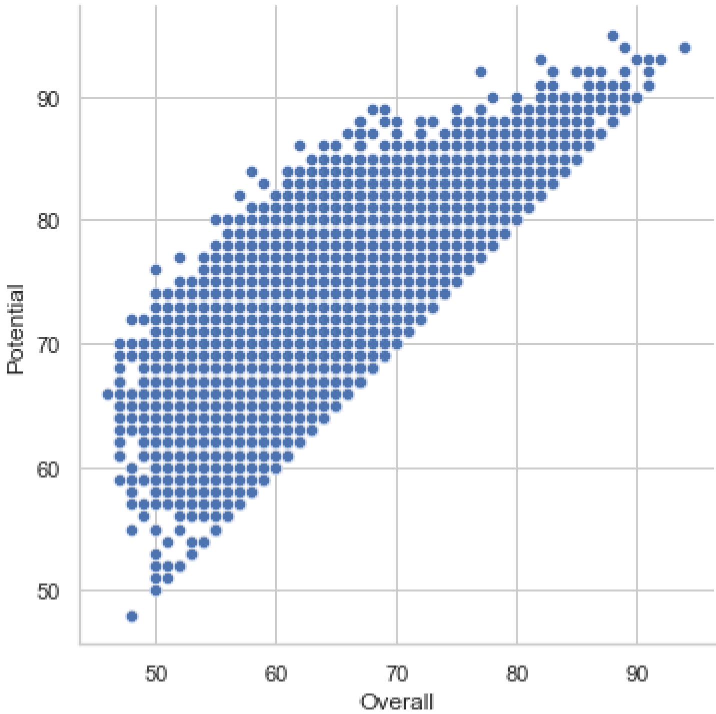
In [58]:
```
#showing standard deviation of observations instead of a confidence interval
f, ax = plt.subplots(figsize=(8, 6))
sns.barplot(x="International Reputation", y="Potential", data=fifa, ci="sd")
plt.show()
```



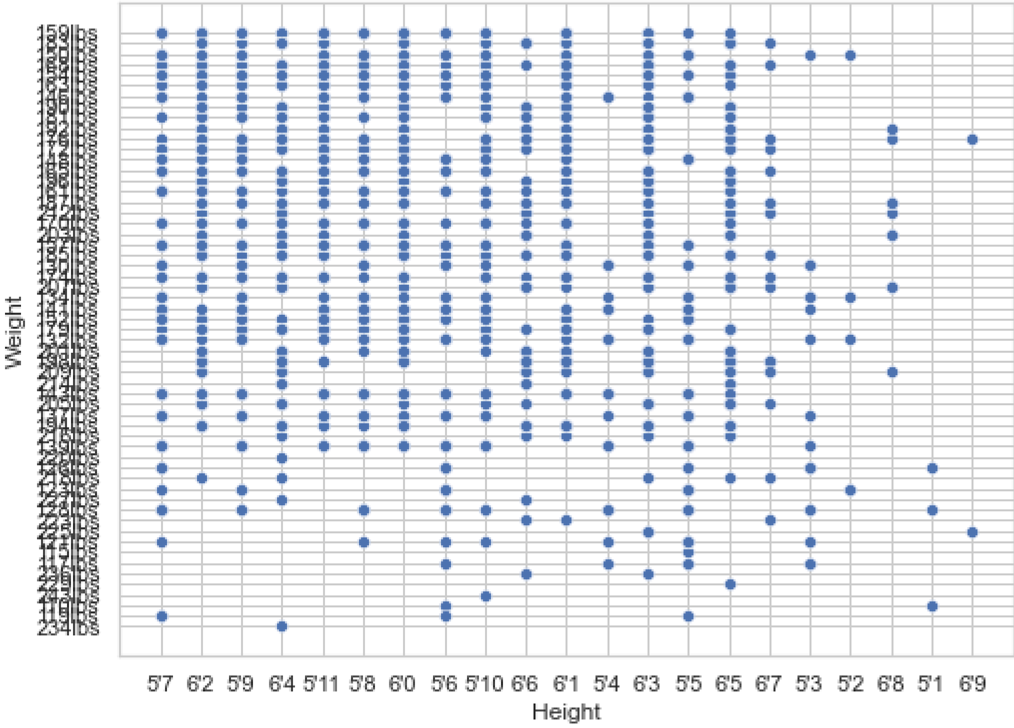
In [59]:
```
#adding “caps” to the error bars
f, ax = plt.subplots(figsize=(8, 6))
sns.barplot(x="International Reputation", y="Potential", data=fifa, capsize=0.2)
plt.show()
```



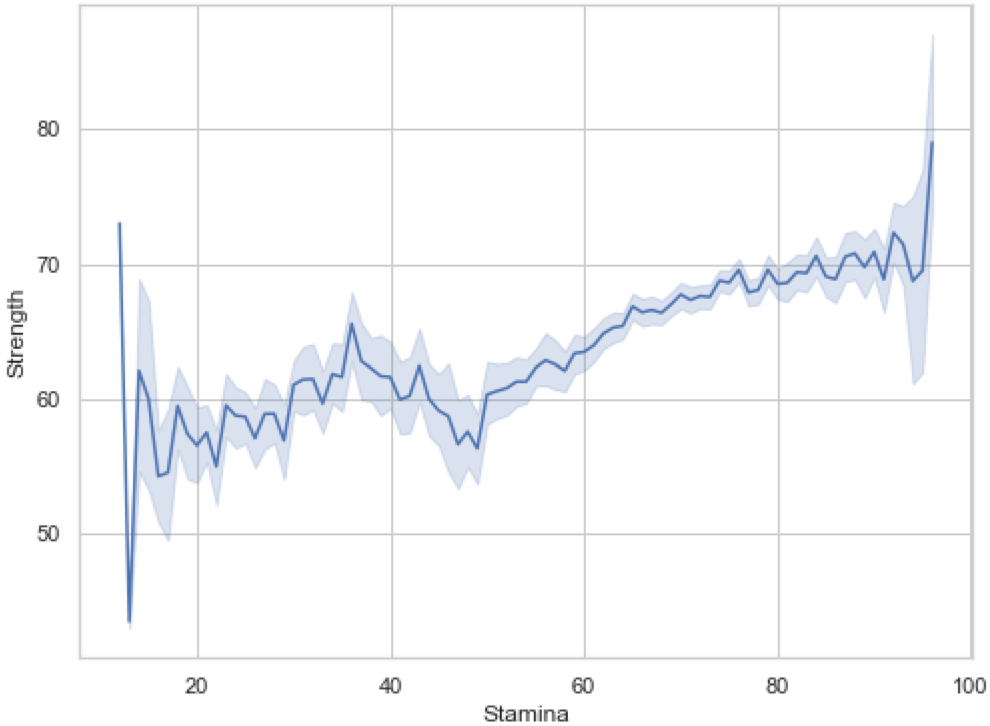
In [60]:
```
#scatterplot with variables Heigh and Weight with Seaborn relplot() function
g = sns.relplot(x="Overall", y="Potential", data=fifa)
```



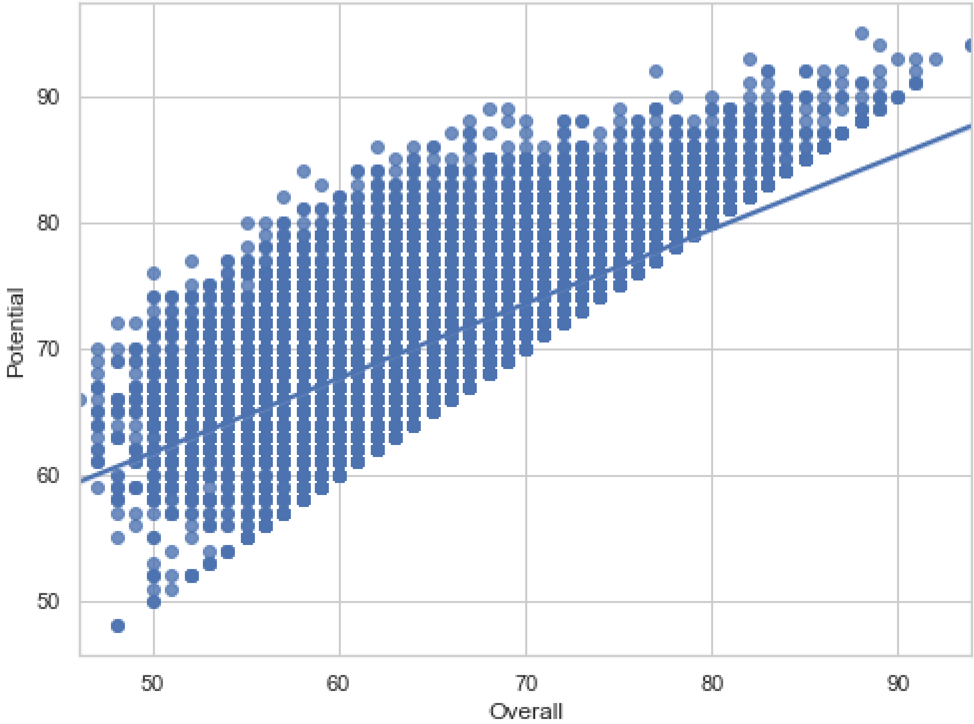
In [61]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.scatterplot(x="Height", y="Weight", data=fifa)
plt.show()
```



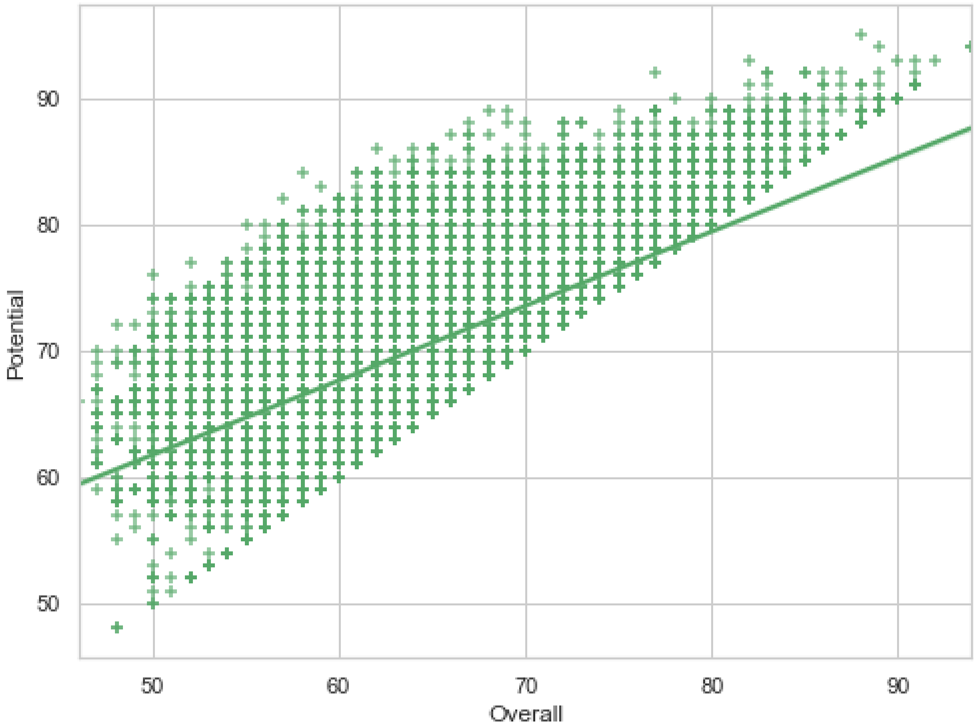
In [63]:
```python
# line plot
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.lineplot(x="Stamina", y="Strength", data=fifa)
plt.show()
```



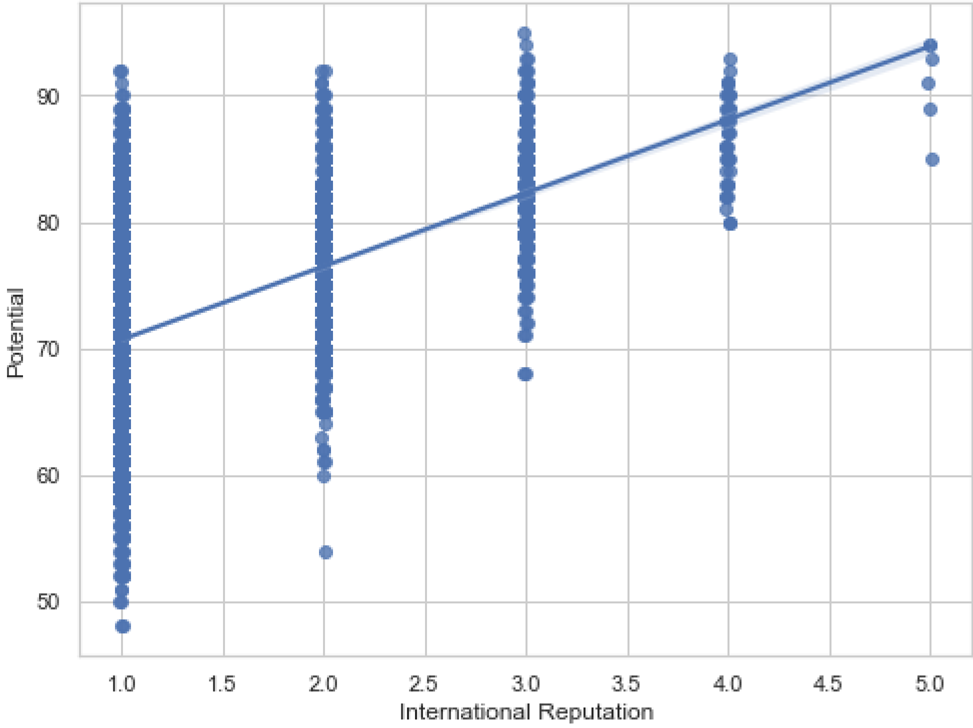
In [64]:
```python
#linear regression model between Overall and Potential variable with regplot() funct
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.regplot(x="Overall", y="Potential", data=fifa)
plt.show()
```



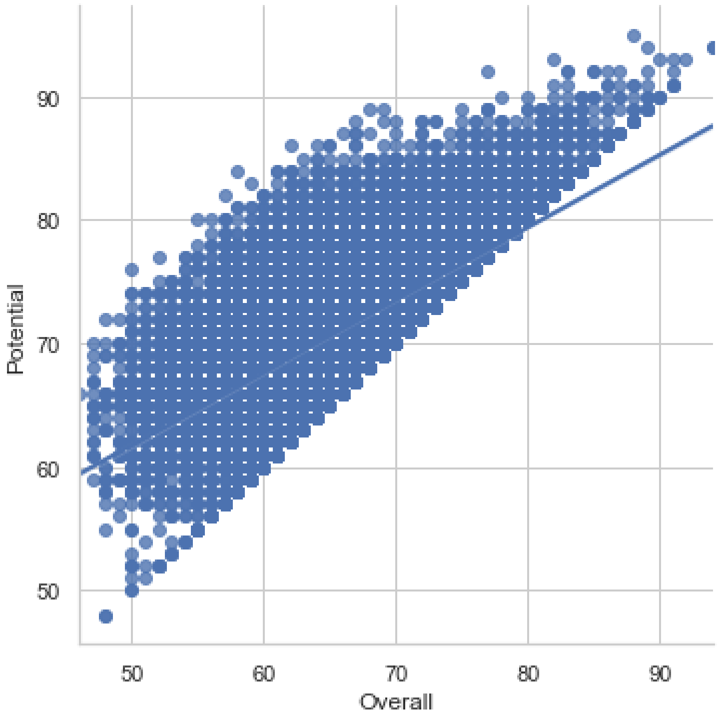
In [65]:
```
#using a different color and marker
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.regplot(x="Overall", y="Potential", data=fifa, color= "g", marker="+")
plt.show()
```



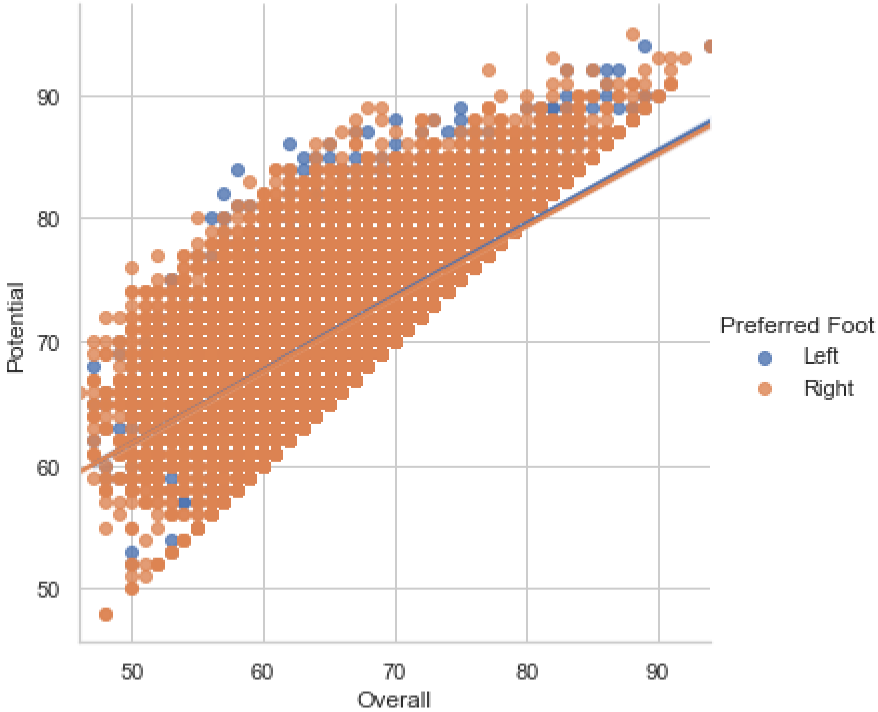
In [66]:
```
#plot with a discrete variable and add some jitter
f, ax = plt.subplots(figsize=(8, 6))
sns.regplot(x="International Reputation", y="Potential", data=fifa, x_jitter=.01)
plt.show()
```



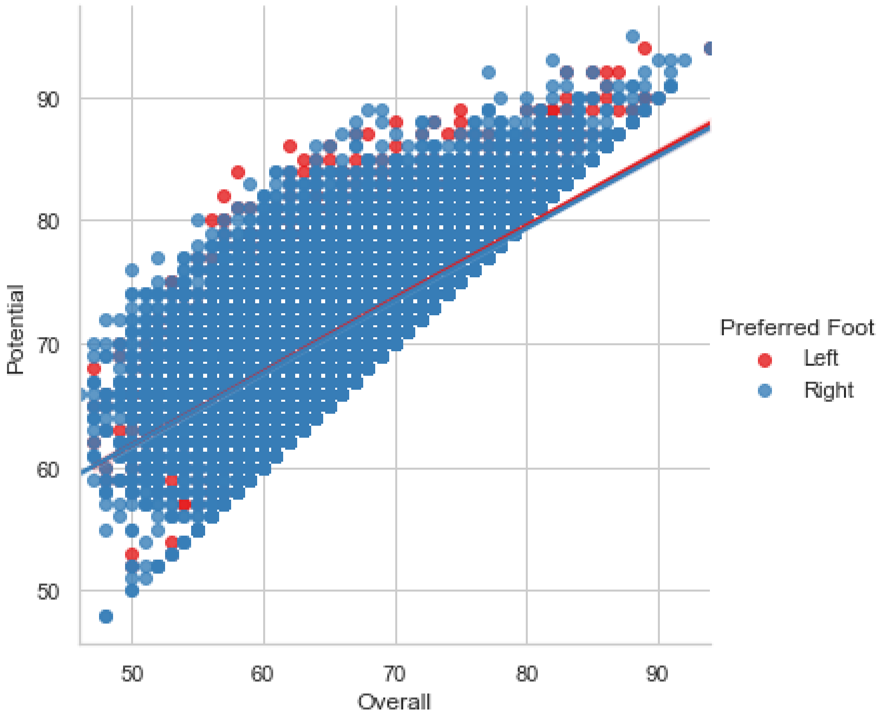
In [67]:
```
#plotting a linear regression model between Overall and Potential variable with lmpl
g= sns.lmplot(x="Overall", y="Potential", data=fifa)
```



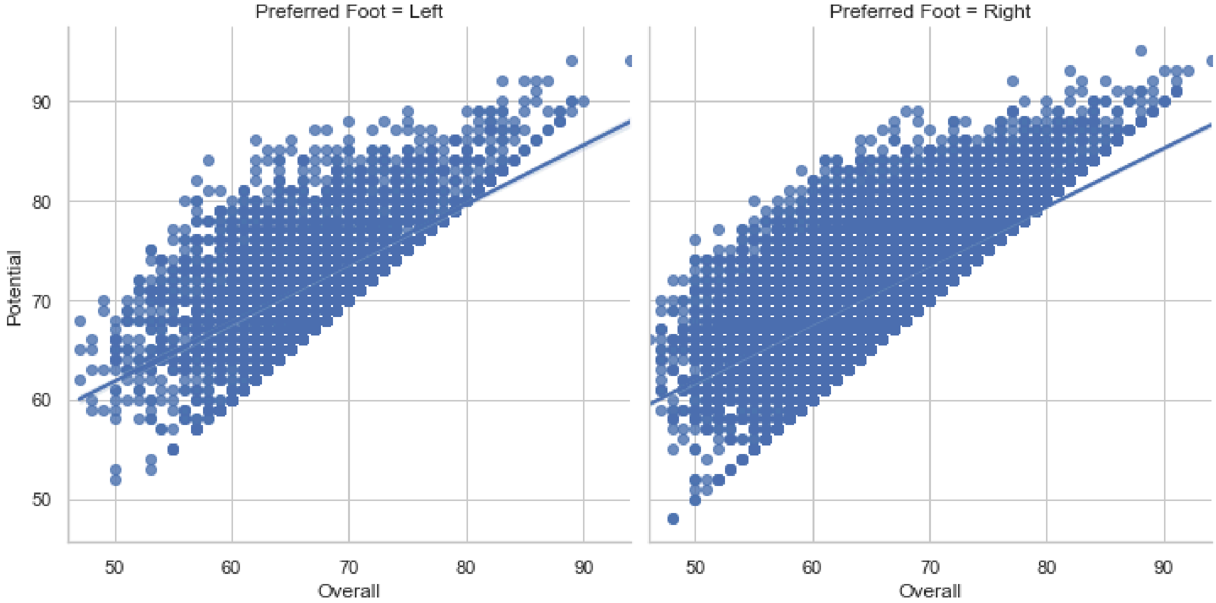
In [68]:
```
#condition on a third variable and plot the levels in different colors
g= sns.lmplot(x="Overall", y="Potential", hue="Preferred Foot", data=fifa)
```



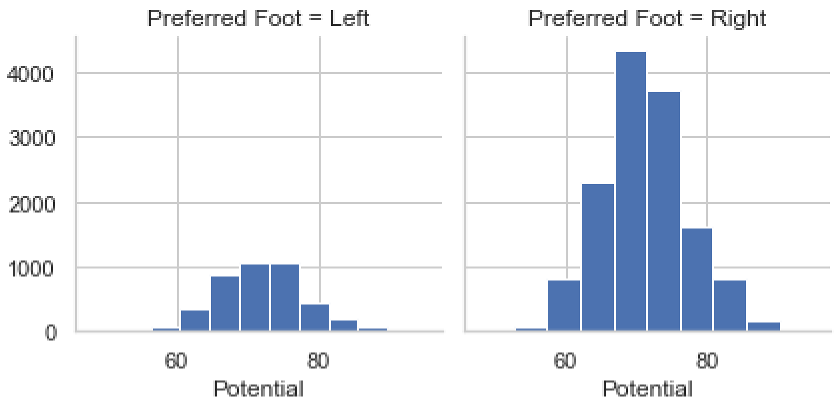
In [69]:
```
#different color palette
g= sns.lmplot(x="Overall", y="Potential", hue="Preferred Foot", data=fifa, palette="
```



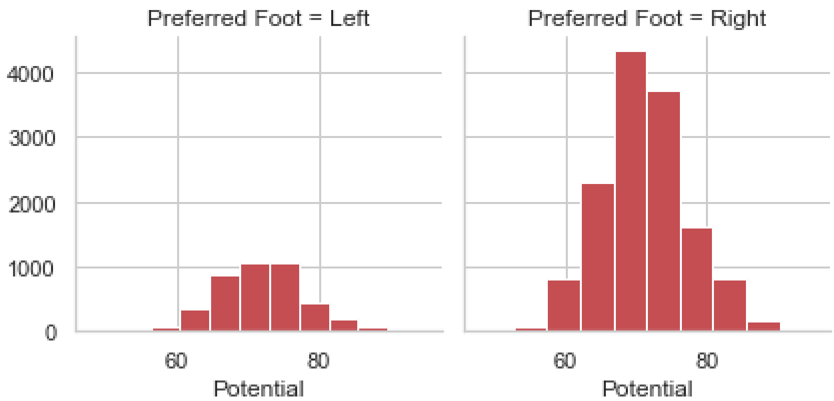
In [70]:
```
#plot the levels of the third variable across different columns
g= sns.lmplot(x="Overall", y="Potential", col="Preferred Foot", data=fifa)
```



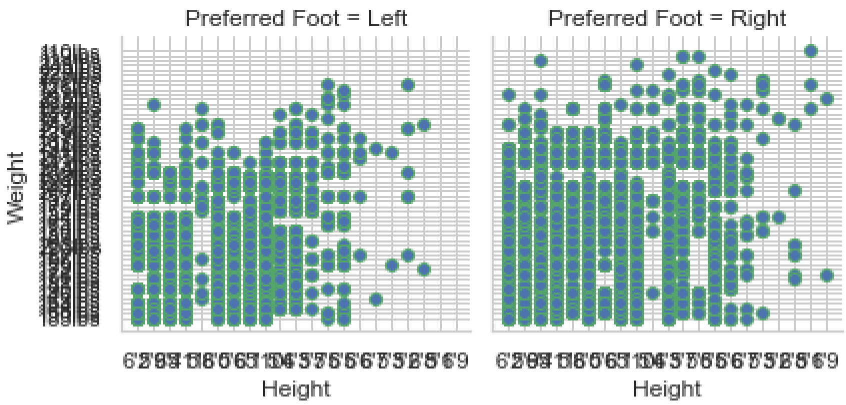
In [71]:
```
#draw a univariate plot of Potential variable on each facet
g = sns.FacetGrid(fifa, col="Preferred Foot")
g = g.map(plt.hist, "Potential")
```



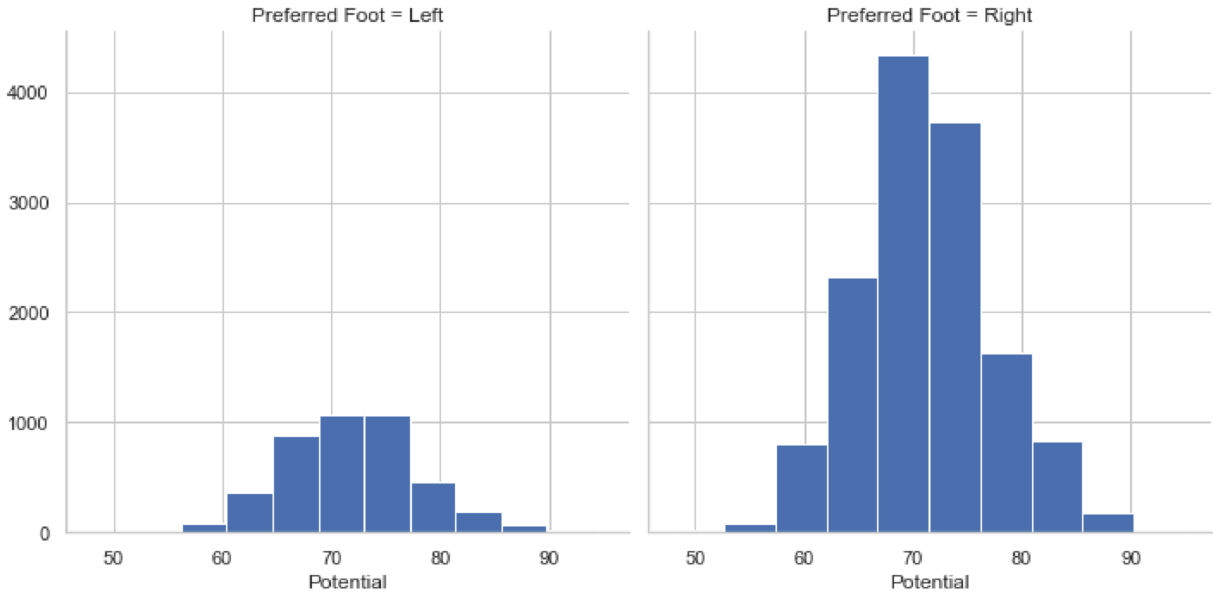
In [72]:
```
g = sns.FacetGrid(fifa, col="Preferred Foot")
g = g.map(plt.hist, "Potential", bins=10, color="r")
```



In [74]:
```
#plot a bivariate function on each facet
g = sns.FacetGrid(fifa, col="Preferred Foot")
g = (g.map(plt.scatter, "Height", "Weight", edgecolor="g").add_legend())
```



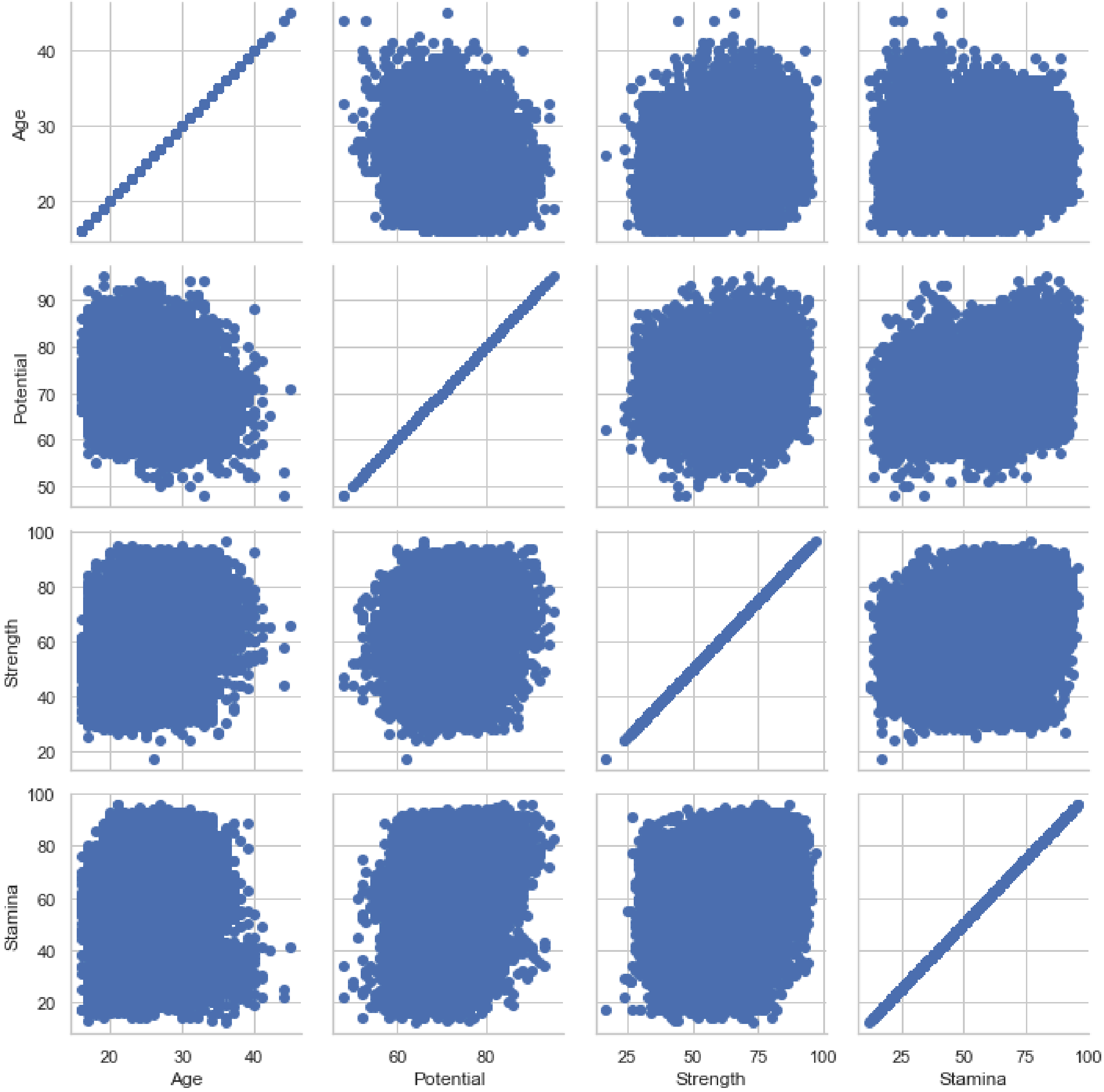
In [75]:
```
g = sns.FacetGrid(fifa, col="Preferred Foot", height=5, aspect=1)
g = g.map(plt.hist, "Potential")
```



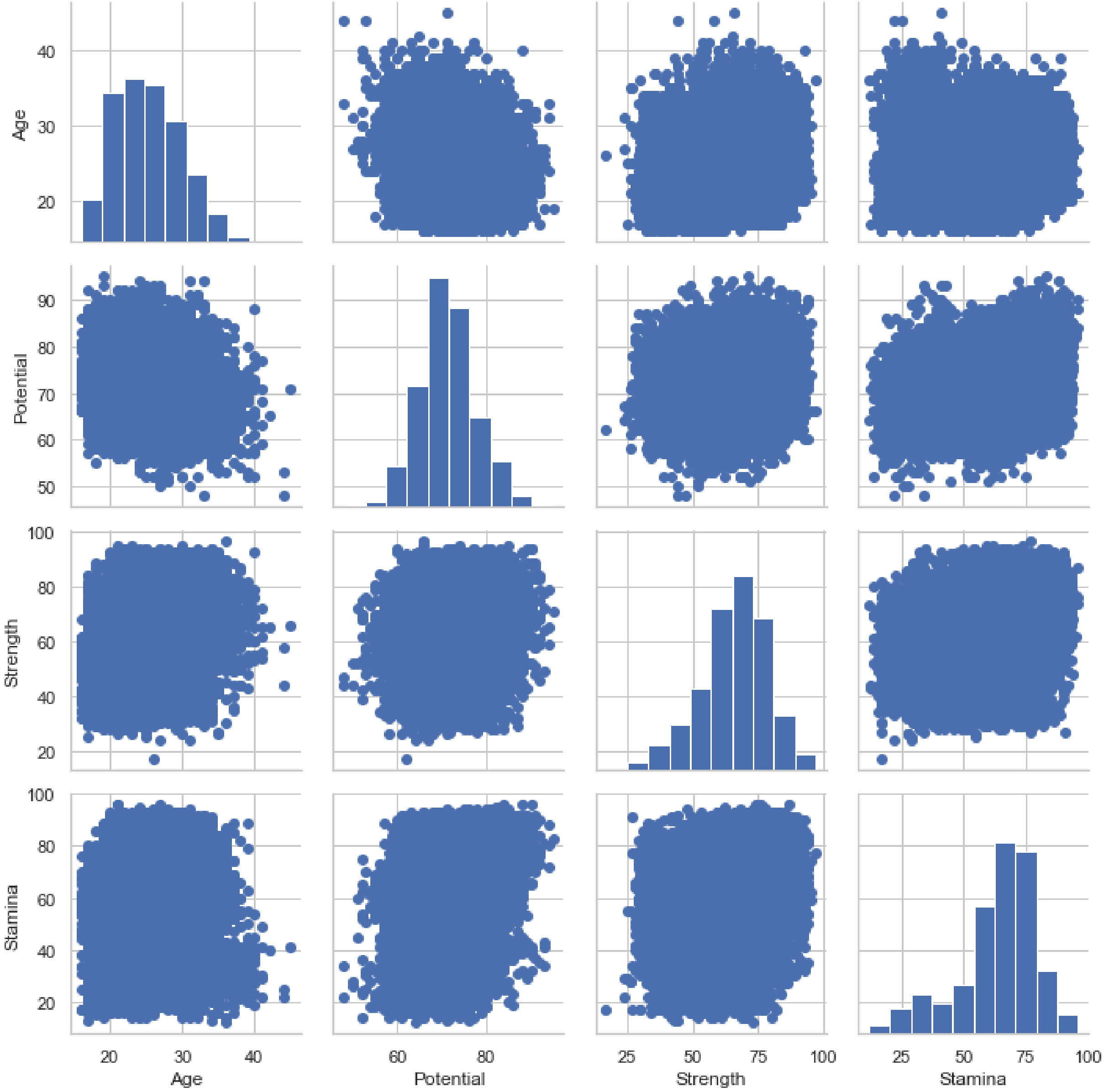
Seaborn `Pairgrid()` function

In [78]:
```
fifa_new = fifa[['Age', 'Potential', 'Strength', 'Stamina', 'Preferred Foot']]
```

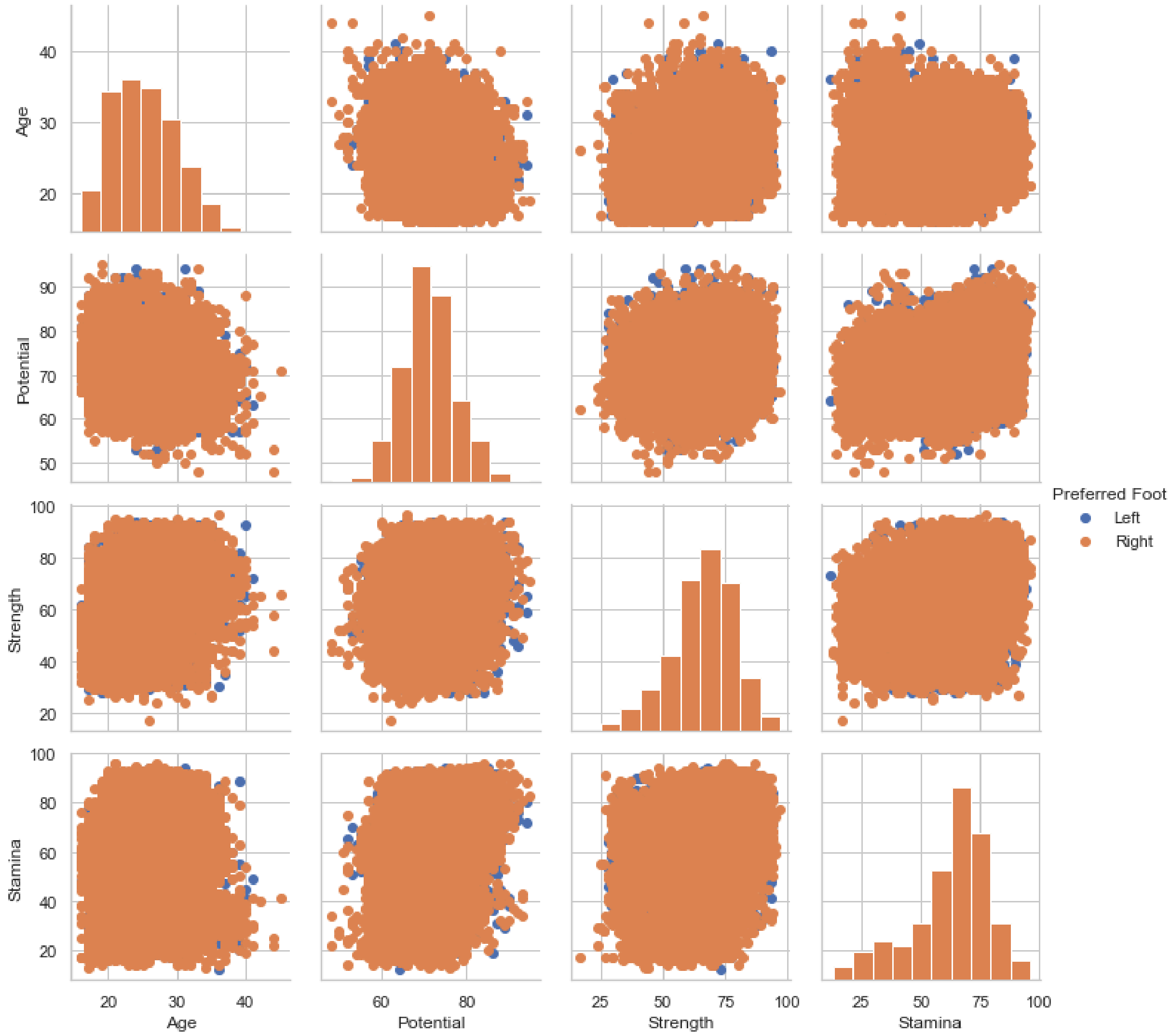
In [79]:
```
g = sns.PairGrid(fifa_new)
g = g.map(plt.scatter)
```



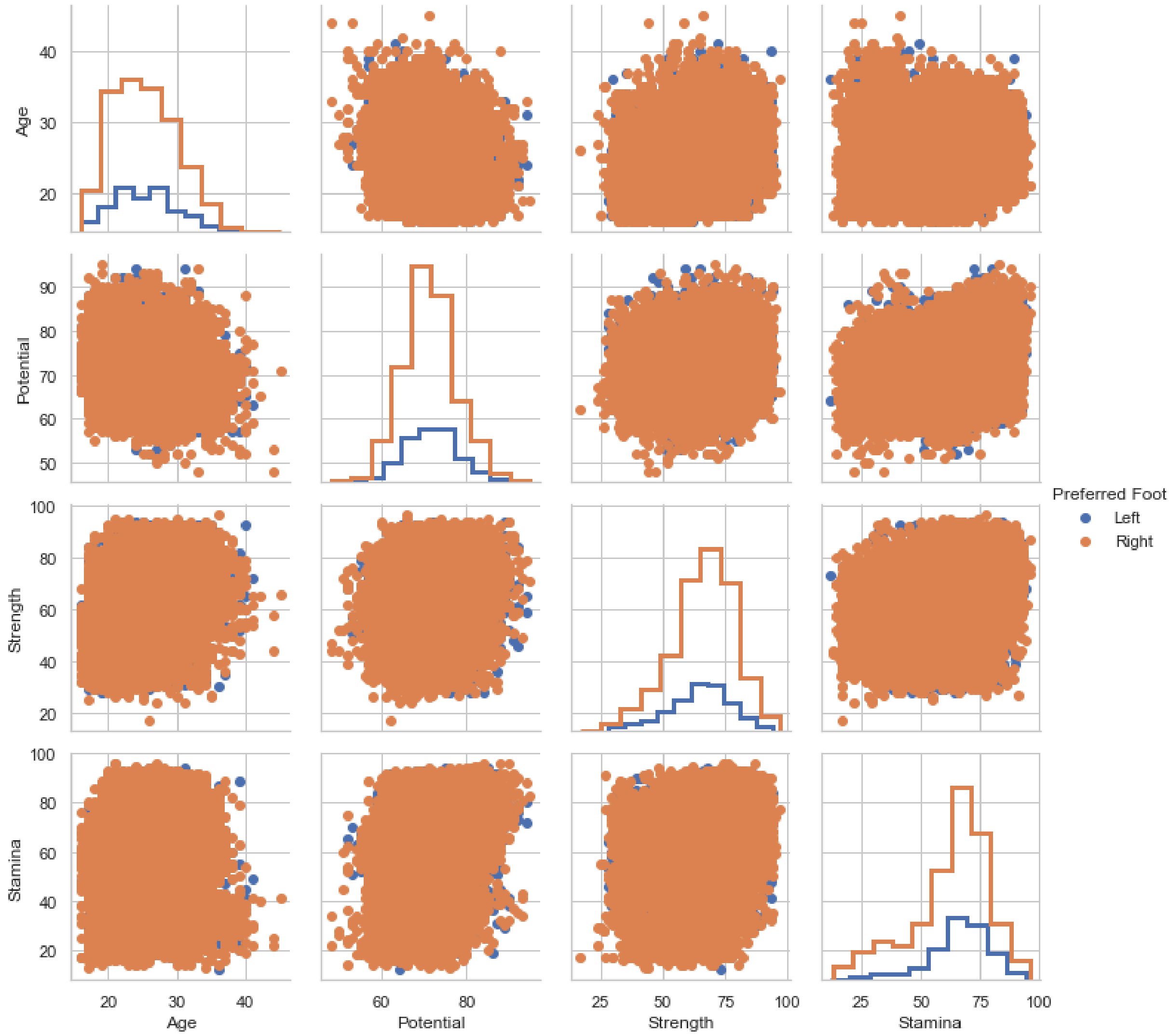
In [80]:
```
#Showing a univariate distribution on the diagonal
g = sns.PairGrid(fifa_new)
g = g.map_diag(plt.hist)
g = g.map_offdiag(plt.scatter)
```



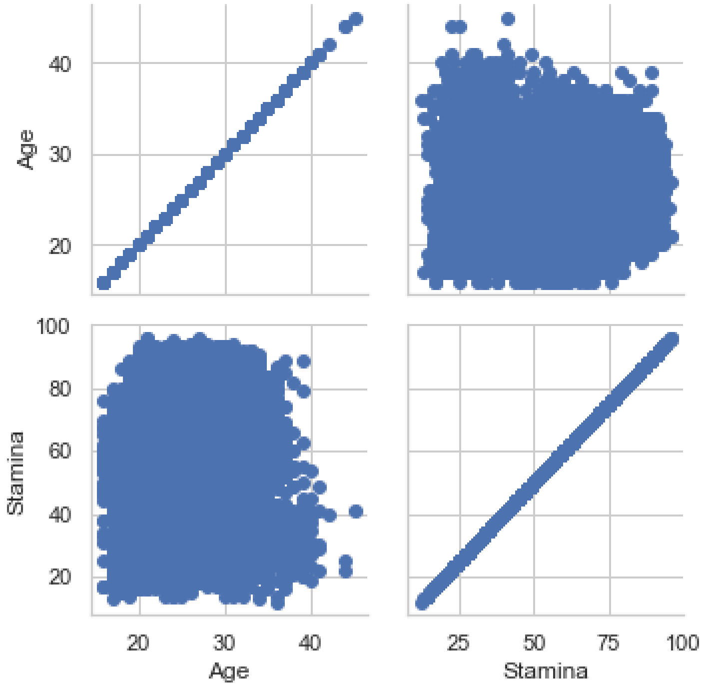
In [81]:
```
#color the points using the categorical variable Preferred Foot
g = sns.PairGrid(fifa_new, hue="Preferred Foot")
g = g.map_diag(plt.hist)
g = g.map_offdiag(plt.scatter)
g = g.add_legend()
```



In [82]:
```
# use a different style to show multiple histograms
g = sns.PairGrid(fifa_new, hue="Preferred Foot")
g = g.map_diag(plt.hist, histtype="step", linewidth=3)
g = g.map_offdiag(plt.scatter)
g = g.add_legend()
```

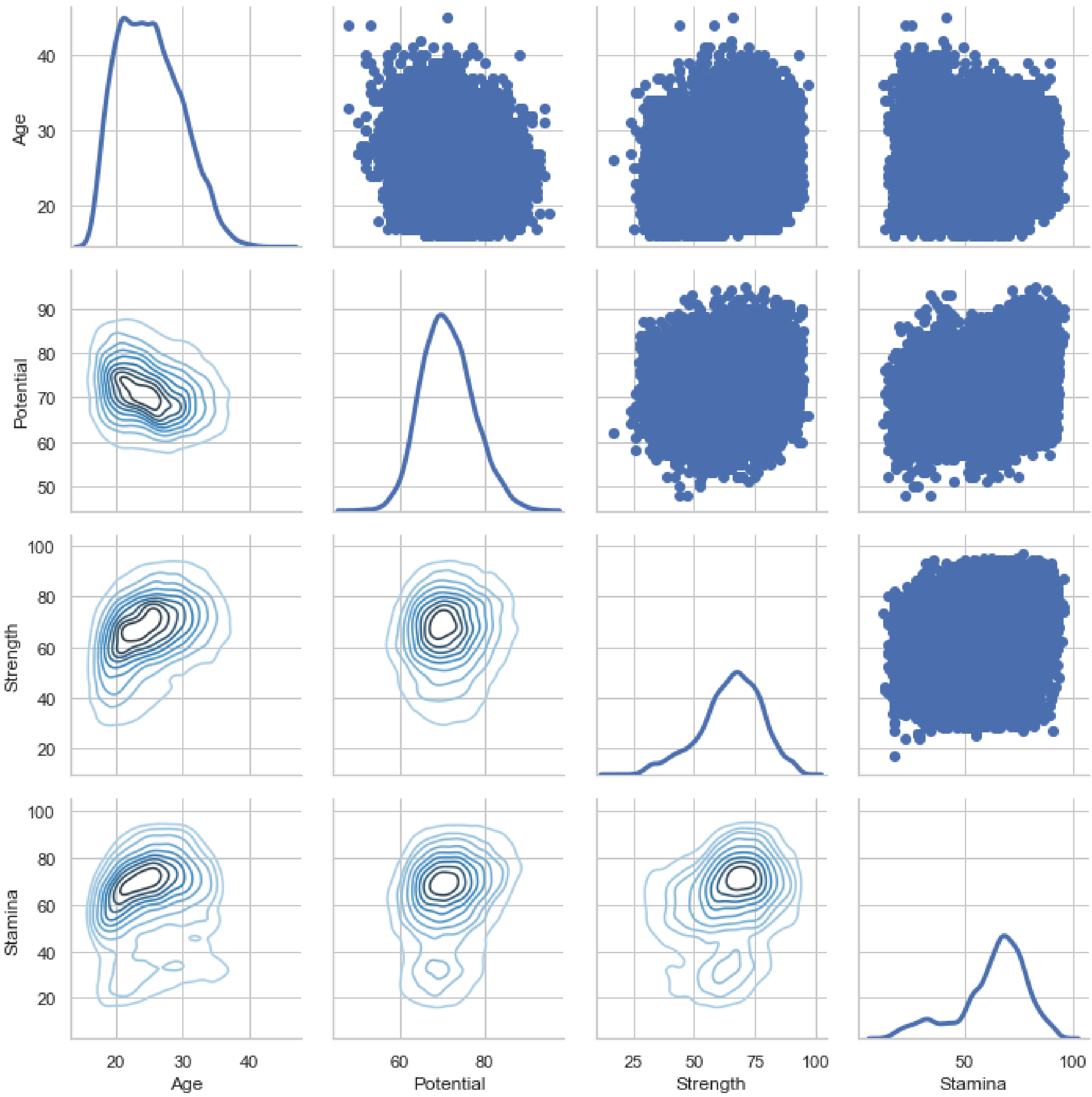


In [90]:
```
#plot a subset of variables
g = sns.PairGrid(fifa_new, vars=['Age', 'Stamina'])
g = g.map(plt.scatter)
```



In [88]:
```
#use different functions on the upper and lower triangles
g = sns.PairGrid(fifa_new)
g = g.map_upper(plt.scatter)
```

```
g = g.map_lower(sns.kdeplot, cmap="Blues_d")
g = g.map_diag(sns.kdeplot, lw=3, legend=False)
```

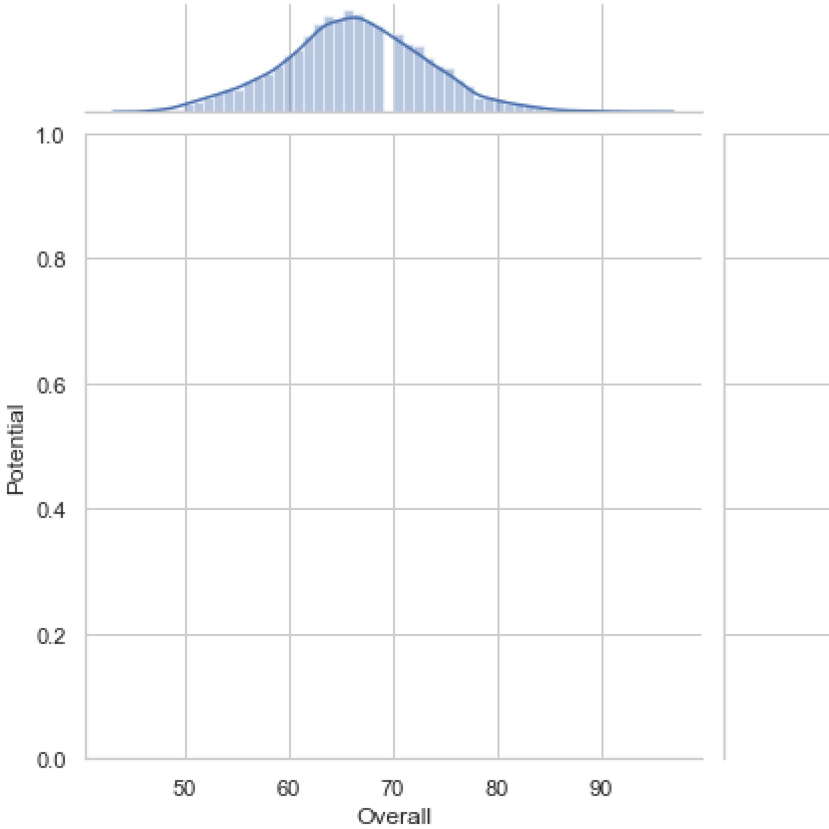


Seaborn Jointgrid() function

In [92]:
```
#initialize the figure and add plots using default parameters
g = sns.JointGrid(x="Overall", y="Potential", data=fifa)
g = g.plot(sns.regplot, sns.distplot)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-92-3c8b233c38f1> in <module>
      1 #initialize the figure and add plots using default parameters
      2 g = sns.JointGrid(x="Overall", y="Potential", data=fifa)
----> 3 g = g.plot(sns.regplot, sns.distplot)

~\anaconda3\lib\site-packages\seaborn\axisgrid.py in plot(self, joint_func, marginal_
func, **kwargs)
   1659 
   1660         """
-> 1661         self.plot_marginals(marginal_func, **kwargs)
   1662         self.plot_joint(joint_func, **kwargs)
   1663         return self

~\anaconda3\lib\site-packages\seaborn\axisgrid.py in plot_marginals(self, func, **kwa
rgs)
```

```
-> 1735         func(y=self.y, **kwargs)
   1733         plt.sca(self.ax_marg_y)
   1734         if str(func.__module__).startswith("seaborn"):
-> 1735             func(y=self.y, **kwargs)
   1736         else:
   1737             func(self.y, vertical=True, **kwargs)

TypeError: distplot() got an unexpected keyword argument 'y'
```



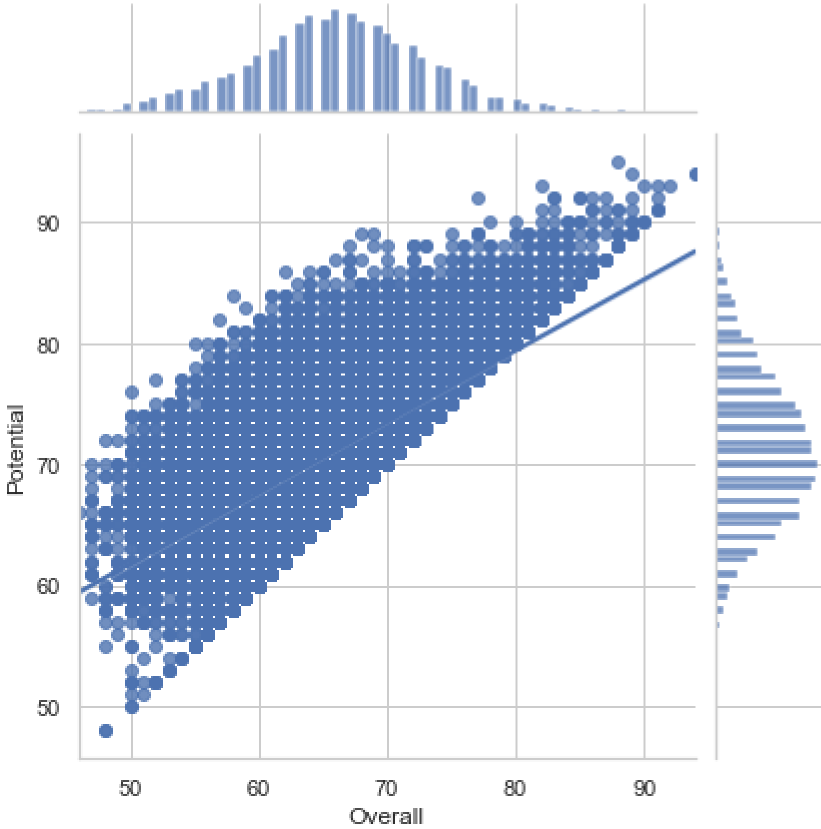
There are a couple of things that need to be updated in the code for it to work correctly:

Update for sns.distplot: In newer versions of Seaborn, sns.distplot has been deprecated and replaced by sns.histplot, sns.kdeplot, or sns.ecdfplot. Depending on the desired type of distribution plot, you would use one of these new functions.

Marginal plots: When using g.plot(), you need to specify the marginal plot functions separately for the horizontal and vertical axes. For the scatter plot with regression line, sns.regplot is fine, but for the distribution plots, you would need to use sns.histplot or another appropriate distribution plot function.

Here's an updated version of the code that addresses these issues and is compatible with Seaborn's more recent versions:

In [95]:
```
# initialize the figure
g = sns.JointGrid(x="Overall", y="Potential", data=fifa)

# plot a regression plot for the bivariate distribution
g = g.plot_joint(sns.regplot)

# plot a distribution plot for the marginal distributions
g = g.plot_marginals(sns.histplot)

plt.show()
```



Here's an explanation of the graph's components:

Central Scatter Plot: The dots represent individual data points and show how 'Potential' values relate to 'Overall' values for different entities (which could be players in a FIFA dataset). The fact that the scatter plot is dense along a diagonal line from the bottom left to the top right suggests a positive correlation: as 'Overall' scores increase, so do 'Potential' scores.

Regression Line: The straight line through the scatter plot is a regression line, which models the linear relationship between 'Overall' and 'Potential'. The upward slope of the line confirms the positive relationship.

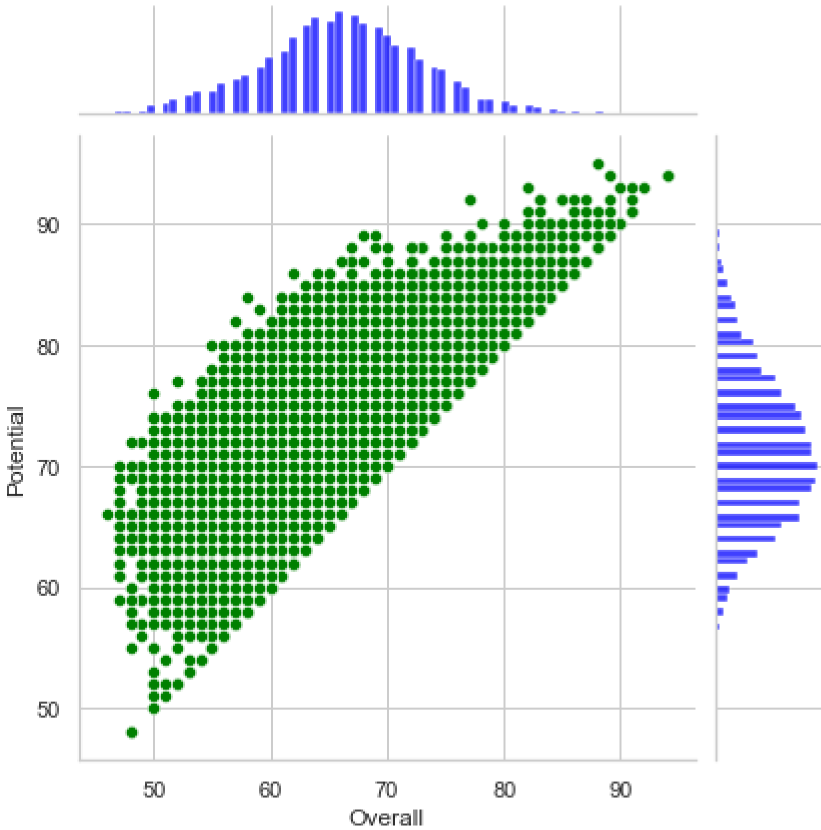
Marginal Histograms: Along the top (x-axis) and the right side (y-axis) of the scatter plot are histograms, which show the distribution of the 'Overall' and 'Potential' values, respectively. The histograms reveal the frequency of observations for different ranges of 'Overall' and 'Potential' scores. In this graph, both distributions appear skewed to the left, meaning there are fewer observations with low 'Overall' and 'Potential' scores and more observations with high scores.

Interpretation:

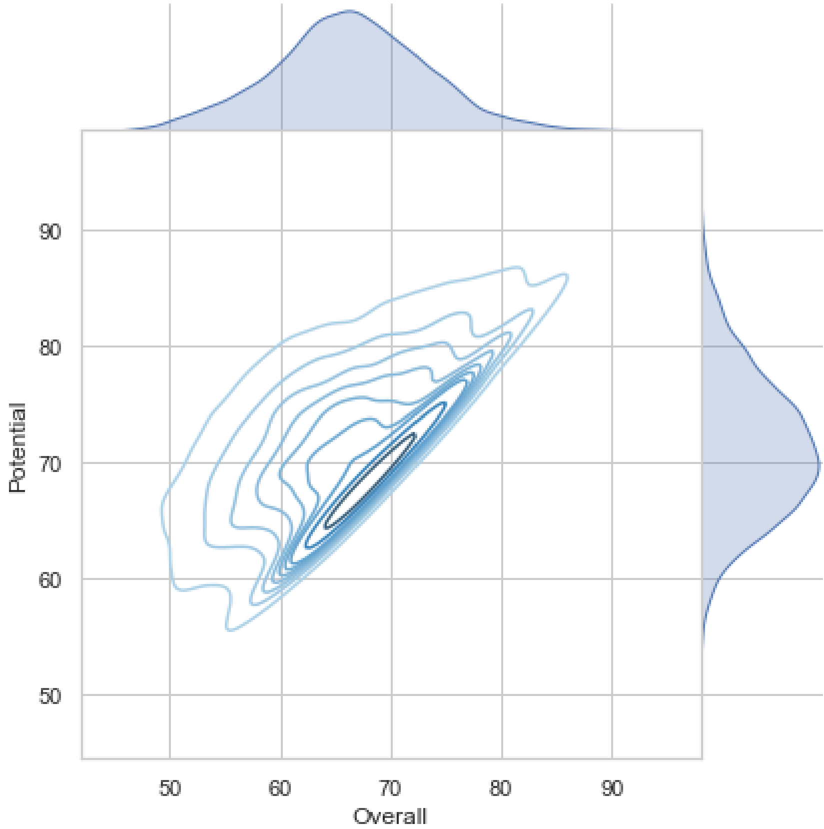
"This graph displays a clear positive trend between 'Overall' and 'Potential', indicating that players with higher current performance levels are also the ones with higher future potential. The concentration of data points along the regression line shows that this trend is consistent across the dataset. The distribution histograms reveal that most players have high 'Overall' and 'Potential' scores, with fewer players at the lower end of the spectrum. This suggests that the dataset likely contains professional players who are already performing at or near their potential."

It's important to note that while the regression line models the relationship between 'Overall' and 'Potential', the density of the scatter plot points around the line indicates that there are variations in this relationship, and not all data points fall exactly on the line. This variability is expected in real-world data and can be influenced by many factors not captured by the two variables plotted.

In [97]:
```
import matplotlib.pyplot as plt

# Assuming fifa is your DataFrame and it has been loaded properly

# Initialize the JointGrid
g = sns.JointGrid(x="Overall", y="Potential", data=fifa)

# Plot a scatterplot in the joint area with a specified color, e.g., 'green'
g = g.plot_joint(sns.scatterplot, color='green')

# Plot histograms on the margins with a specified color, e.g., 'blue'
g = g.plot_marginals(sns.histplot, color='blue')

# Display the plot
plt.show()
```

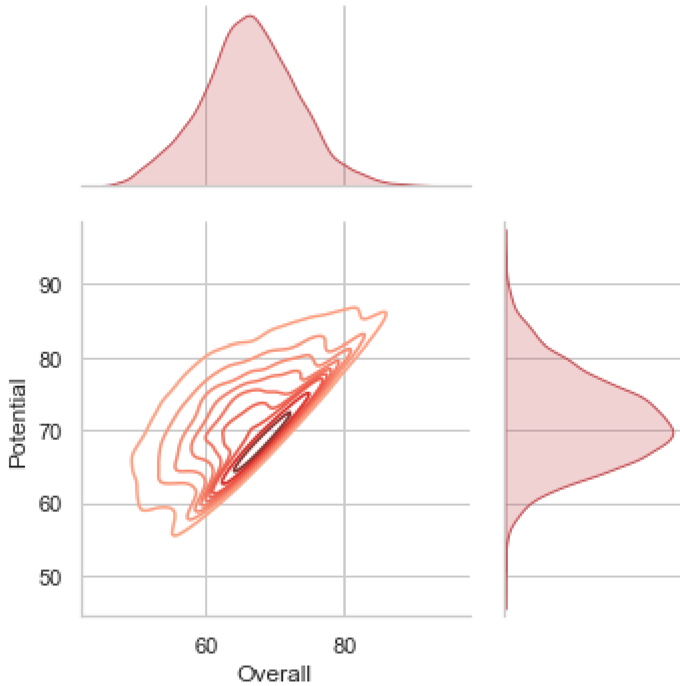


In [98]:
```
#remove the space between the joint and marginal axes
g = sns.JointGrid(x="Overall", y="Potential", data=fifa, space=0)
g = g.plot_joint(sns.kdeplot, cmap="Blues_d")
g = g.plot_marginals(sns.kdeplot, shade=True)
```



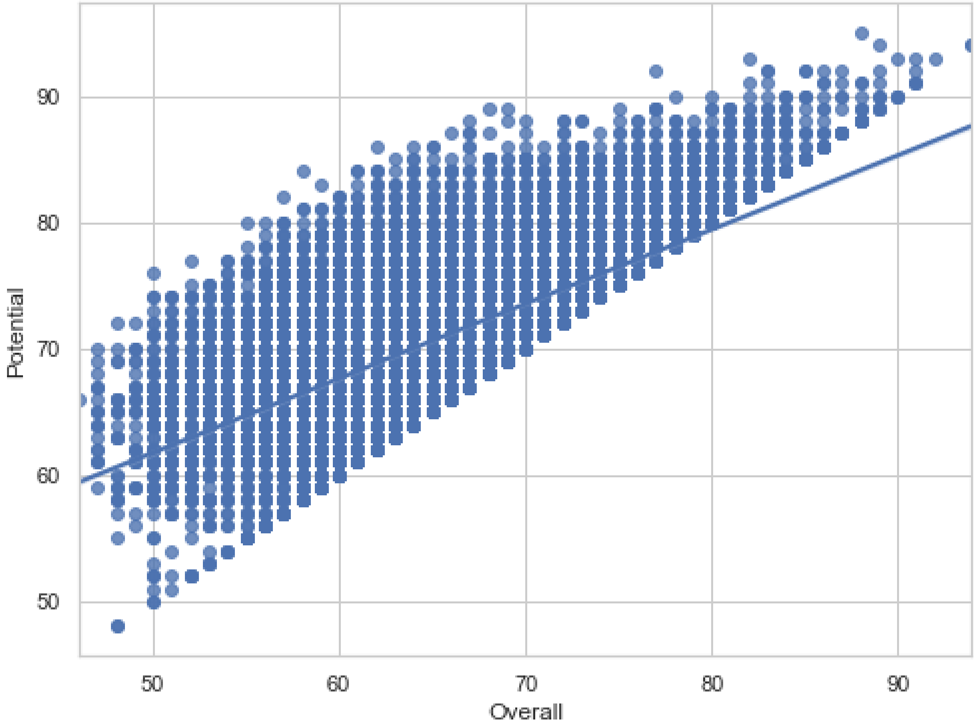
draw a smaller plot with relatively larger marginal axes

In [99]:
```
g = sns.JointGrid(x="Overall", y="Potential", data=fifa, height=5, ratio=2)
g = g.plot_joint(sns.kdeplot, cmap="Reds_d")
g = g.plot_marginals(sns.kdeplot, color="r", shade=True)
```



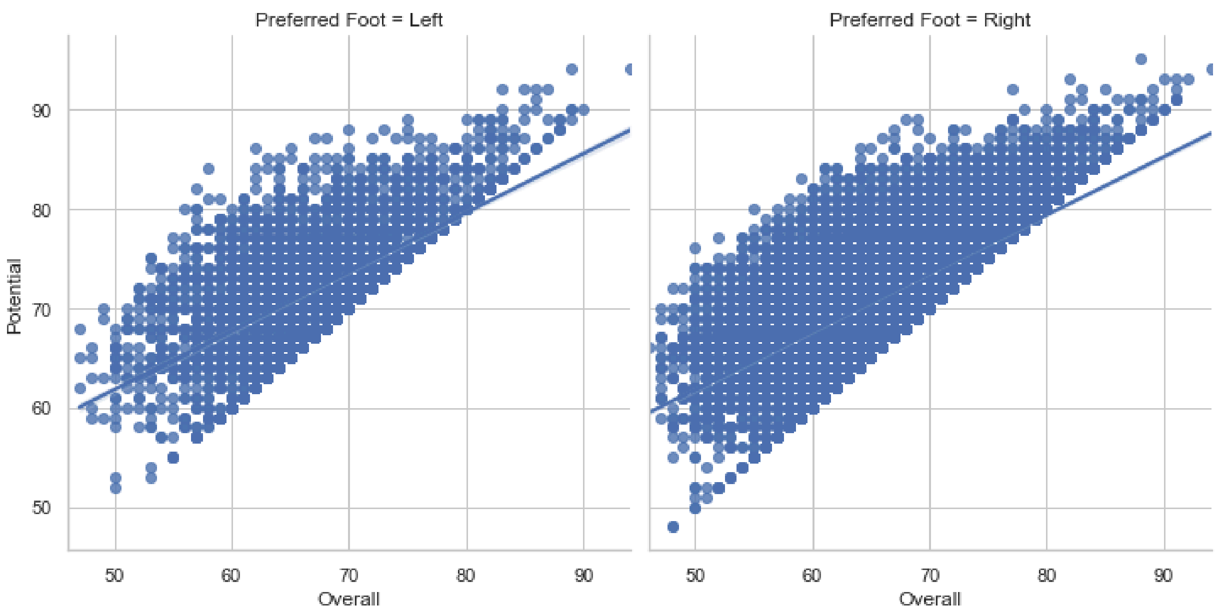
Controlling the size and shape of the plot

In [100…
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.regplot(x="Overall", y="Potential", data=fifa);
```



In [101…
```
sns.lmplot(x="Overall", y="Potential", col="Preferred Foot", data=fifa, col_wrap=2,
```

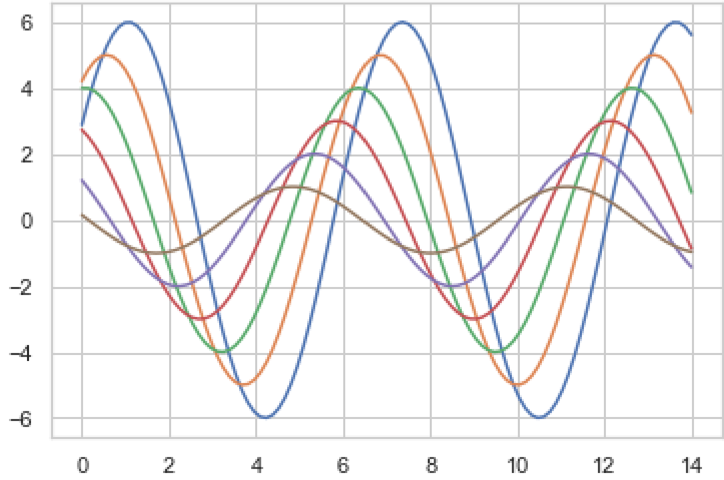
Out[101…
```
<seaborn.axisgrid.FacetGrid at 0x262907c1c10>
```



Seaborn figure styles

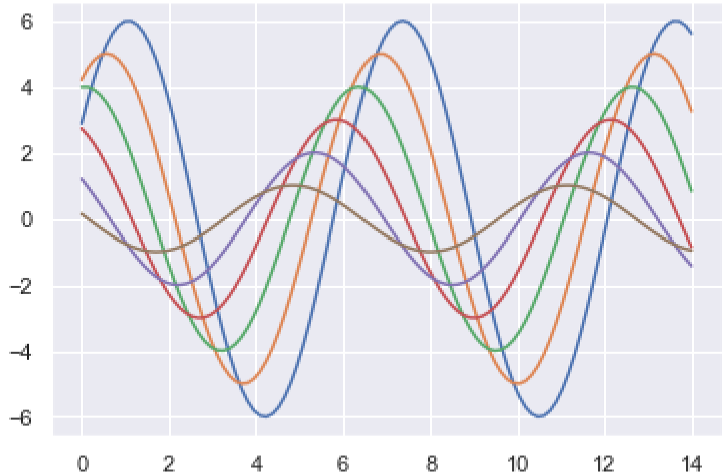
In [103…
```
def sinplot(flip=1):
    x = np.linspace(0, 14, 100)
    for i in range(1, 7):
        plt.plot(x, np.sin(x + i * .5) * (7 - i) * flip)
```

In [104…
```
#matplotlib default parameters.
sinplot()
```



In [105…

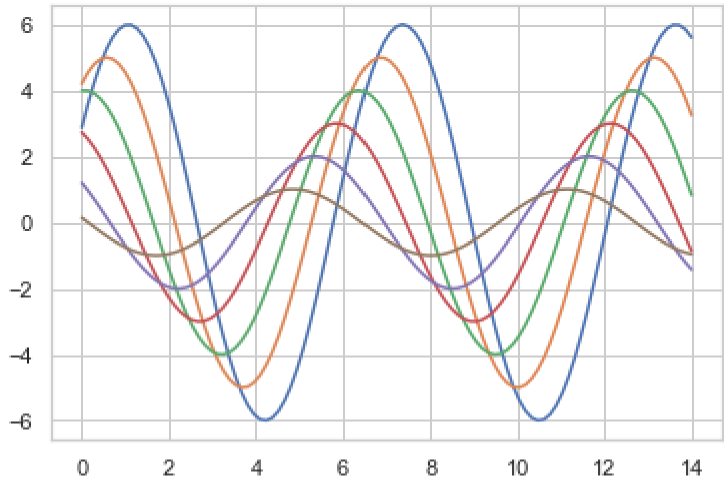
```
#seaborn defaults
sns.set()
sinplot()
```



set different styles

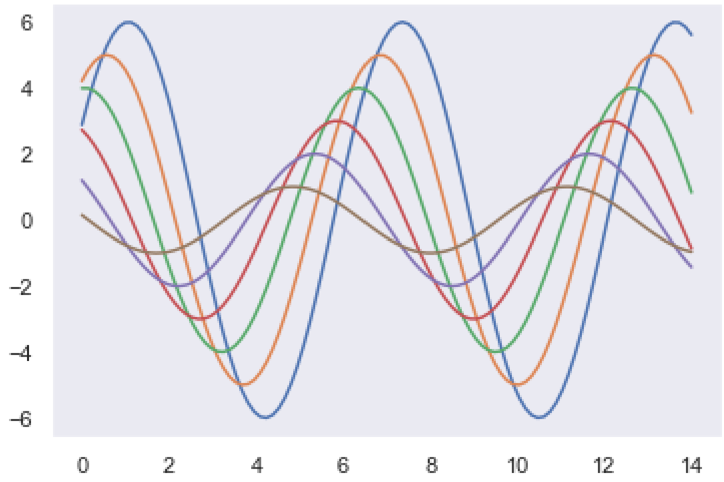
In [106…

```
sns.set_style("whitegrid")
sinplot()
```



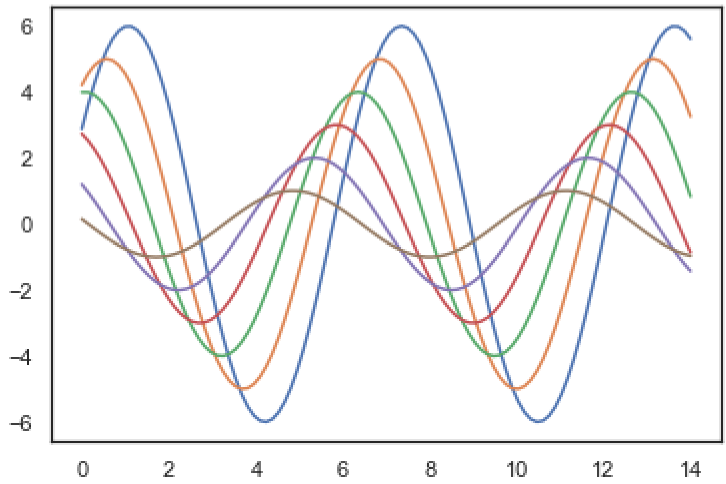
In [107…

```
sns.set_style("dark")
sinplot()
```



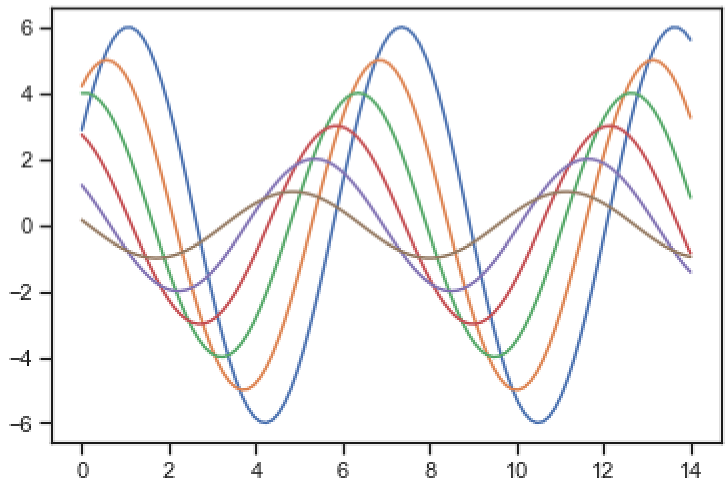
In [108…

```
sns.set_style("white")
sinplot()
```



In [ ]:

```
sns.set_style("ticks")
sinplot()
```



In [ ]: