Data Visualization for FIFA dataset using Seaborn

In [2]:

```
# importing libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set(style='whitegrid')
from collections import Counter
```

In [3]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [4]:

```
# importing the dataset

fifa19=pd.read_csv(r"C:\Users\Jan Saida\OneDrive\Documents\Desktop\Excel sheets\FIFA.csv")
fifa19
```

Out[4]:

	Unnamed: 0	ID	Name	Age	Photo	Nationality	Flag	Overall	Potential	Club	...	Composure	Marking	StandingTac
0	0	158023	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	Argentina	https://cdn.sofifa.org/flags/52.png	94	94	FC Barcelona	...	96.0	33.0	2
1	1	20801	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	Portugal	https://cdn.sofifa.org/flags/38.png	94	94	Juventus	...	95.0	28.0	3
2	2	190871	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	Brazil	https://cdn.sofifa.org/flags/54.png	92	93	Paris Saint-Germain	...	94.0	27.0	2
3	3	193080	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	Spain	https://cdn.sofifa.org/flags/45.png	91	93	Manchester United	...	68.0	15.0	2
4	4	192985	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	Belgium	https://cdn.sofifa.org/flags/7.png	91	92	Manchester City	...	88.0	68.0	5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
18202	18202	238813	J. Lundstram	19	https://cdn.sofifa.org/players/4/19/238813.png	England	https://cdn.sofifa.org/flags/14.png	47	65	Crewe Alexandra	...	45.0	40.0	4
18203	18203	243165	N. Christoffersson	19	https://cdn.sofifa.org/players/4/19/243165.png	Sweden	https://cdn.sofifa.org/flags/46.png	47	63	Trelleborgs FF	...	42.0	22.0	1
18204	18204	241638	B. Worman	16	https://cdn.sofifa.org/players/4/19/241638.png	England	https://cdn.sofifa.org/flags/14.png	47	67	Cambridge United	...	41.0	32.0	1
18205	18205	246268	D. Walker-Rice	17	https://cdn.sofifa.org/players/4/19/246268.png	England	https://cdn.sofifa.org/flags/14.png	47	66	Tranmere Rovers	...	46.0	20.0	2
18206	18206	246269	G. Nugent	16	https://cdn.sofifa.org/players/4/19/246269.png	England	https://cdn.sofifa.org/flags/14.png	46	66	Tranmere Rovers	...	43.0	40.0	4

18207 rows × 89 columns

EDA - Exploratory Data Analysis

In [6]:

```
# preview of dataset

fifa19.head()
```

Out[6]:

	Unnamed: 0	ID	Name	Age	Photo	Nationality	Flag	Overall	Potential	Club	...	Composure	Marking	StandingTackle	Slidin
0	0	158023	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	Argentina	https://cdn.sofifa.org/flags/52.png	94	94	FC Barcelona	...	96.0	33.0	28.0	
1	1	20801	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	Portugal	https://cdn.sofifa.org/flags/38.png	94	94	Juventus	...	95.0	28.0	31.0	
2	2	190871	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	Brazil	https://cdn.sofifa.org/flags/54.png	92	93	Paris Saint-Germain	...	94.0	27.0	24.0	
3	3	193080	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	Spain	https://cdn.sofifa.org/flags/45.png	91	93	Manchester United	...	68.0	15.0	21.0	
4	4	192985	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	Belgium	https://cdn.sofifa.org/flags/7.png	91	92	Manchester City	...	88.0	68.0	58.0	

5 rows × 89 columns

In [7]:

```
# Summary/info of the dataset

fifa19.info()
```

```
<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]:
```python
fifa19['Body Type'].value_counts()
```

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

Comment

- This dataset contains 89 variables.

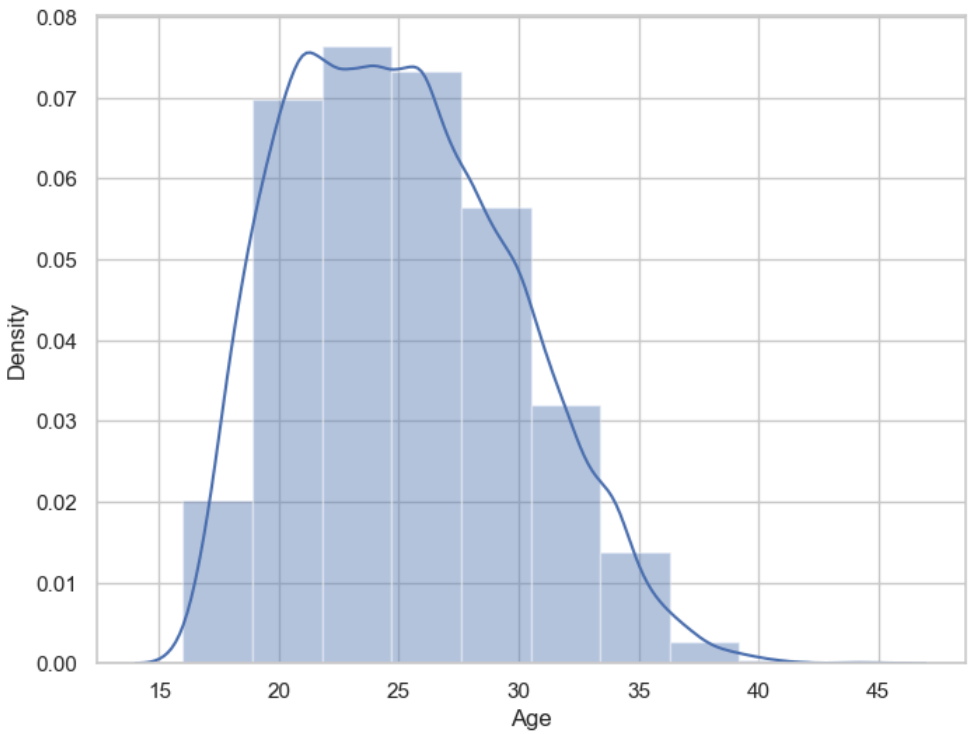
- Out of the 89 variables, 44 are numerical variables. 38 are of float64 data type and remaining 6 are of int64 data type.
- The remaining 45 variables are of character data type.
- Let's explore this further.

Exploring age variable

Visualize distribution of `Age` variable with Seaborn `distplot()` function

- Seaborn `distplot()` function flexibly plots a univariate distribution of observations.
- This function combines the matplotlib hist function (with automatic calculation of a good default bin size) with the seaborn kdeplot() and rugplot() functions.
- So, let's visualize the distribution of `Age` variable with Seaborn `distplot()` function.

In [11]:
```
f,ax=plt.subplots(figsize=(8,6))
x=fifa19['Age']
ax=sns.distplot(x,bins=10)
plt.show()
```

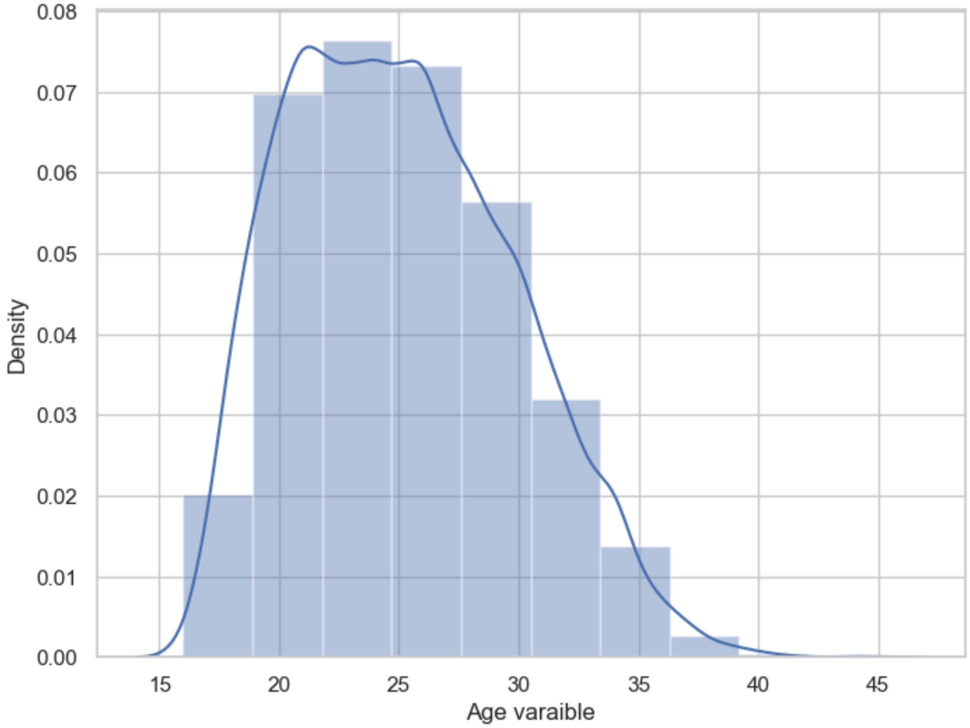


Comment

- It can be seen that the `Age` variable is slightly positively skewed.

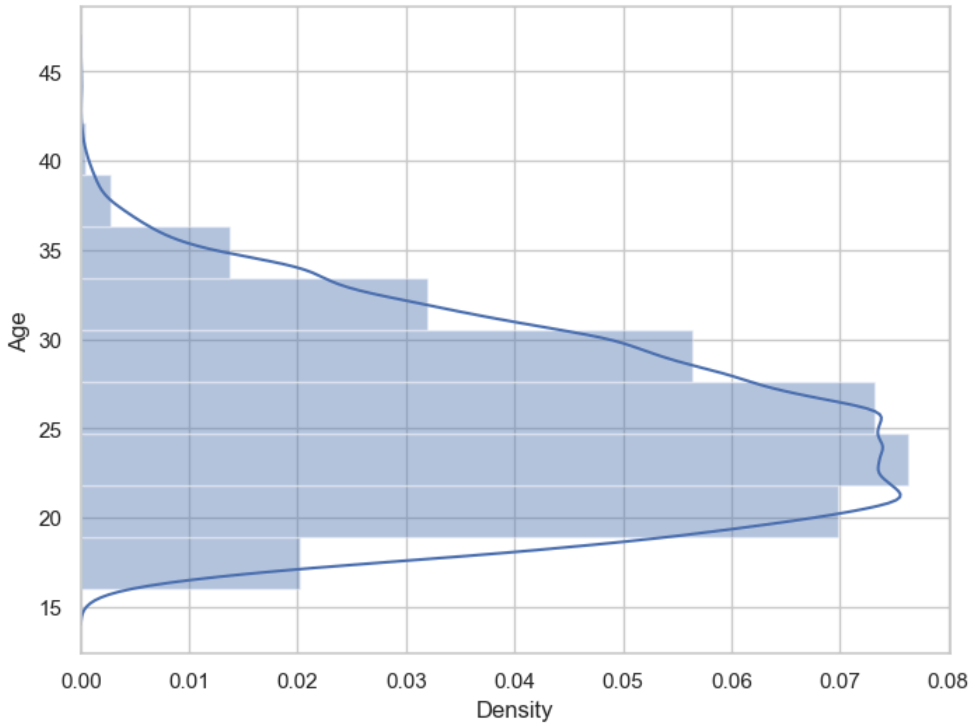
We can use Pandas series object to get an informative axis label as follows-

In [14]:
```
f,ax=plt.subplots(figsize=(8,6))
x=fifa19['Age']
x=pd.Series(x,name='Age varaible')
ax=sns.distplot(x,bins=10)
plt.show()
```



we can plot the distribution on the vertical axis as follows:-

In [16]:
```
f, ax = plt.subplots(figsize=(8,6))
x = fifa19['Age']
ax = sns.distplot(x, bins=10, vertical = True)
plt.show()
```

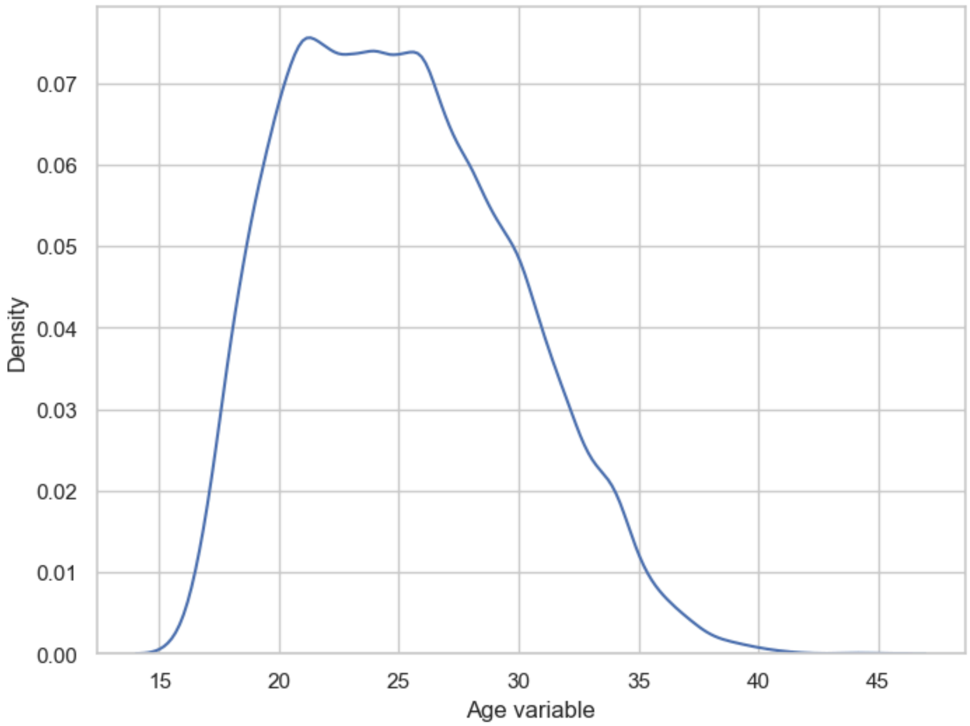


Seaborn Kernel Density Estimation (KDE) Plot

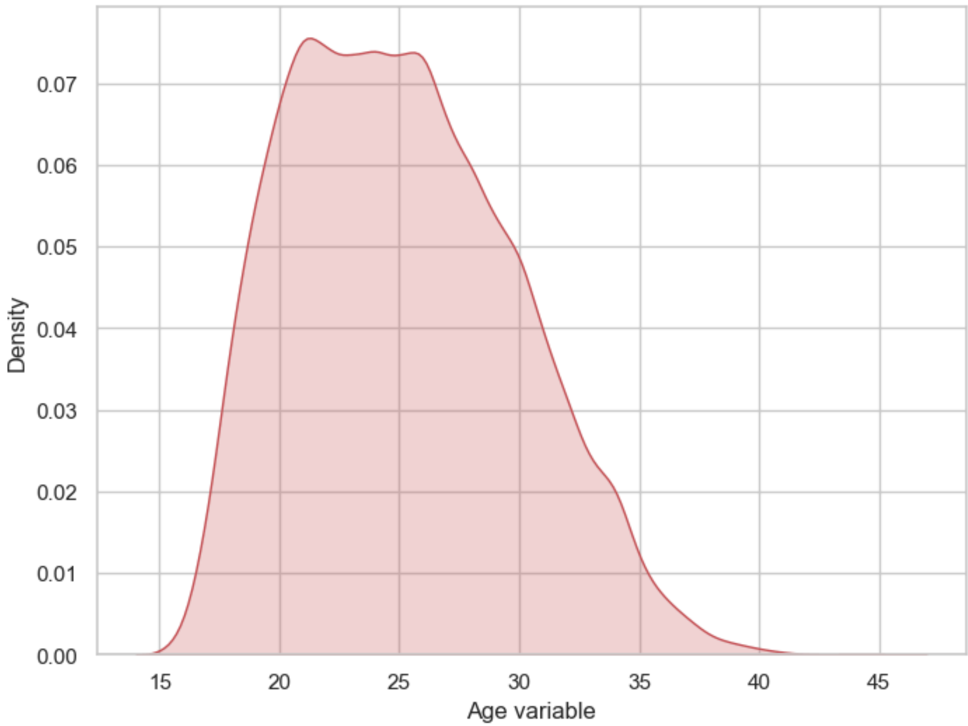
- The `kernel density estimate (KDE)` plot is a useful tool for plotting the shape of a distribution.
- Seaborn kdeplot is another seaborn plotting function that fits and plot a univariate or bivariate kernel density estimate.
- Like the histogram, the KDE plots encode the density of observations on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

In [18]:

```
f, ax = plt.subplots(figsize=(8,6))
x = fifa19['Age']
x = pd.Series(x, name="Age variable")
ax = sns.kdeplot(x)
plt.show()
```



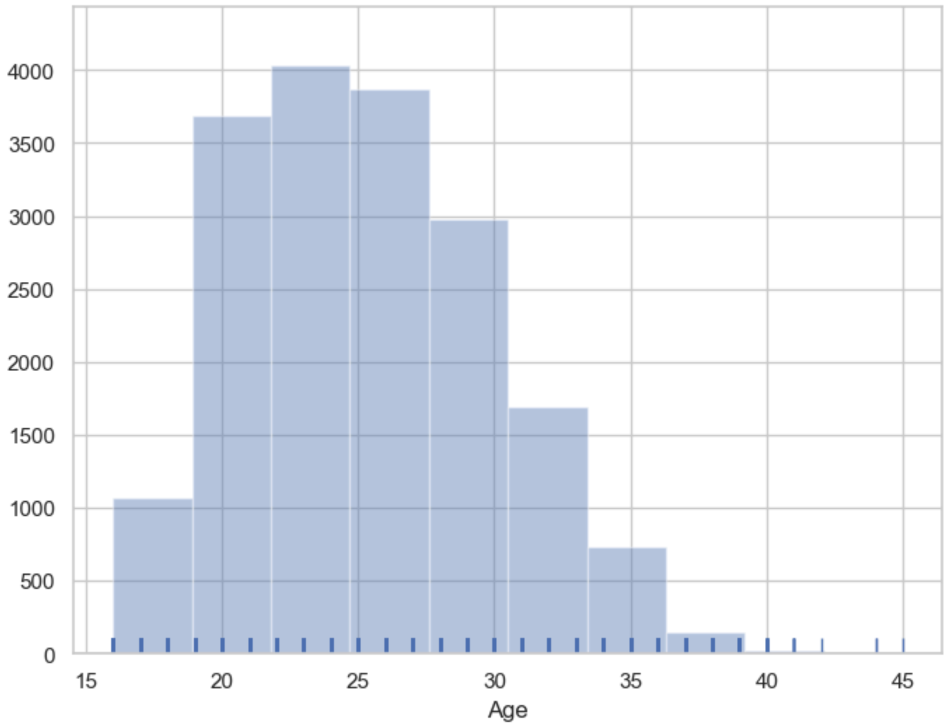
In [19]:
```
# we can shade under the density curve and use a different colors as follows:-

f, ax = plt.subplots(figsize=(8,6))
x = fifa19['Age']
x = pd.Series(x, name="Age variable")
ax = sns.kdeplot(x, shade=True, color='r')
plt.show()
```



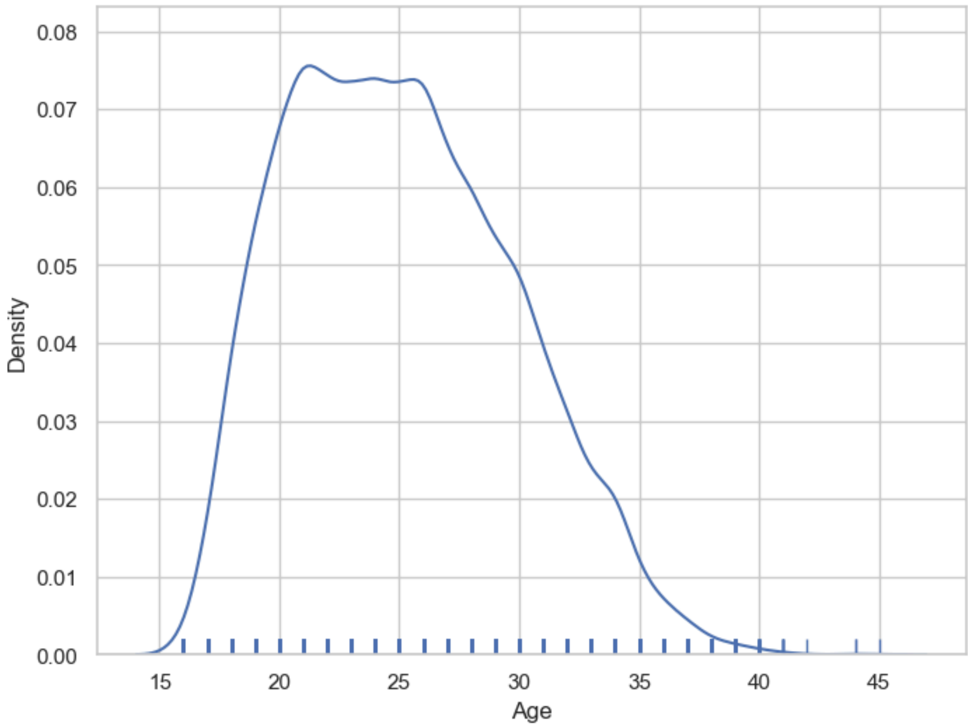
Histograms

- A histogram represents the distribution of data by forming bins along the range of the data and then drawing bars to show the number of observations that fall in each bin.
- A `hist()` function already exists in matplotlib.
- We can use Seaborn to plot a histogram.

In [21]:
```
f, ax = plt.subplots(figsize=(8,6))
x = fifa19['Age']
ax = sns.distplot(x, kde=False, rug=True, bins=10)
plt.show()
```



In [22]:
```
# we can plot a KDE plot alternatively as follows:-

f, ax = plt.subplots(figsize=(8,6))
x = fifa19['Age']
ax = sns.distplot(x, hist=False, rug=True, bins=10)
plt.show()
```



Explore `Preferred Foot` variable

In [24]:
```
# Checking number of unique values in `Preferred Foot` variable

fifa19['Preferred Foot'].nunique()
```

Out[24]:
```
2
```

In [25]:
```
# Checking frequency distribution of values in `Preferred Foot` variable

fifa19['Preferred Foot'].value_counts()
```

Out[25]:
```
Preferred Foot
Right    13948
Left      4211
Name: count, dtype: int64
```

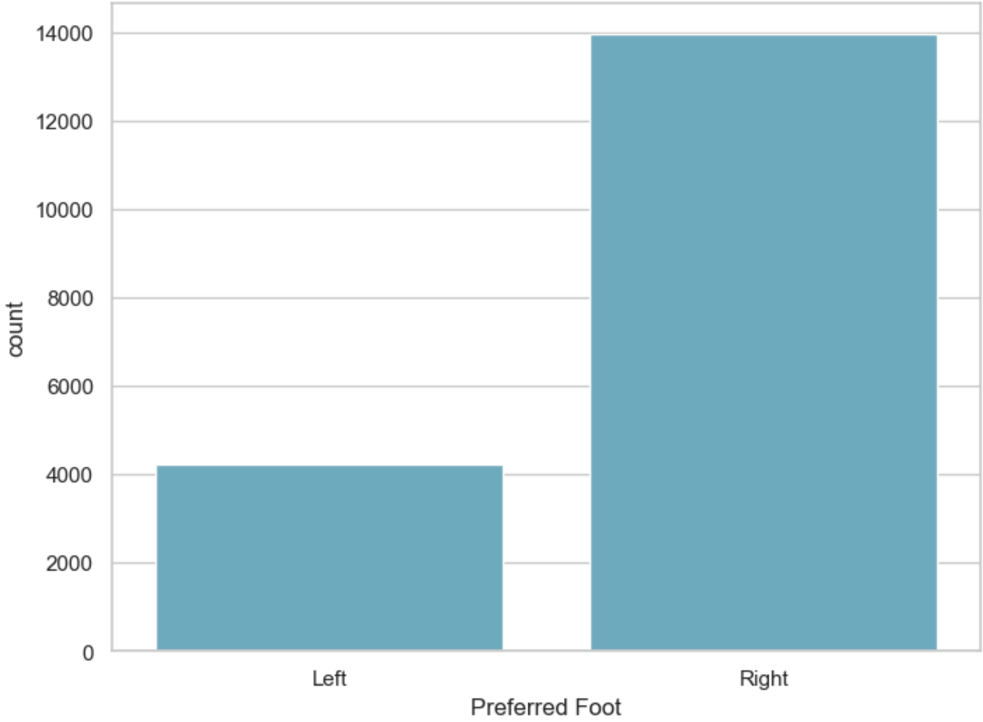
Visualize distribution of values with Seaborn `countplot()` function.

- A countplot shows the counts of observations in each categorical bin using bars.
- It can be thought of as a histogram across a categorical, instead of quantitative, variable.

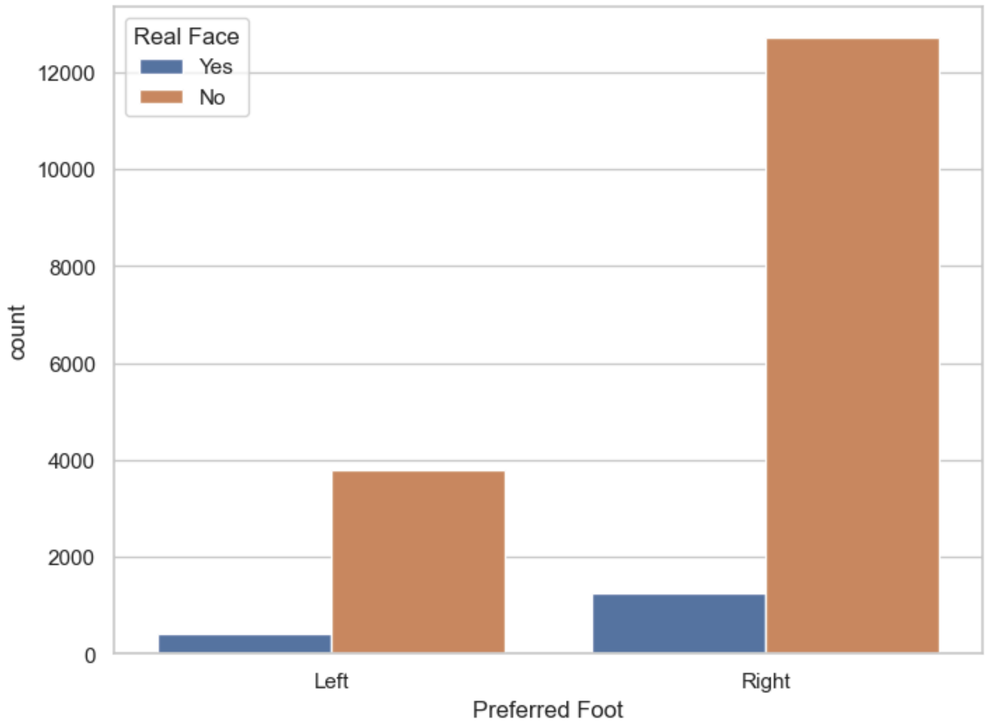
- This function always treats one of the variables as categorical and draws data at ordinal positions (0, 1, … n) on the relevant axis, even when the data has a numeric or date type.

1. - We can visualize the distribution of values with Seaborn `countplot()` function as follows-

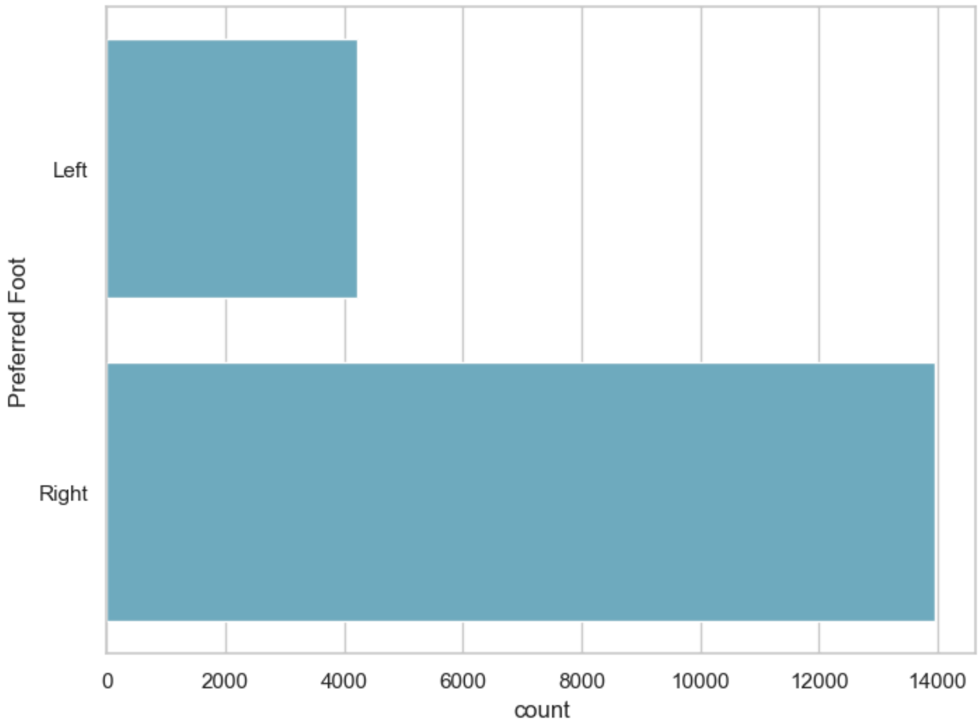
In [27]:
```
f,ax=plt.subplots(figsize=(8,6))
sns.countplot(x='Preferred Foot',data=fifa19,color='c')
plt.show()
```



In [28]:
```
# we can show value counts for two categorical varibale as follows:-

f, ax =plt.subplots(figsize=(8,6))
sns.countplot(x='Preferred Foot',hue='Real Face',data=fifa19)
plt.show()
```



In [29]:
```
# we can draw plot vertically as follows:-

f,ax=plt.subplots(figsize=(8,6))
sns.countplot(y='Preferred Foot',data=fifa19,color='c')
plt.show()
```

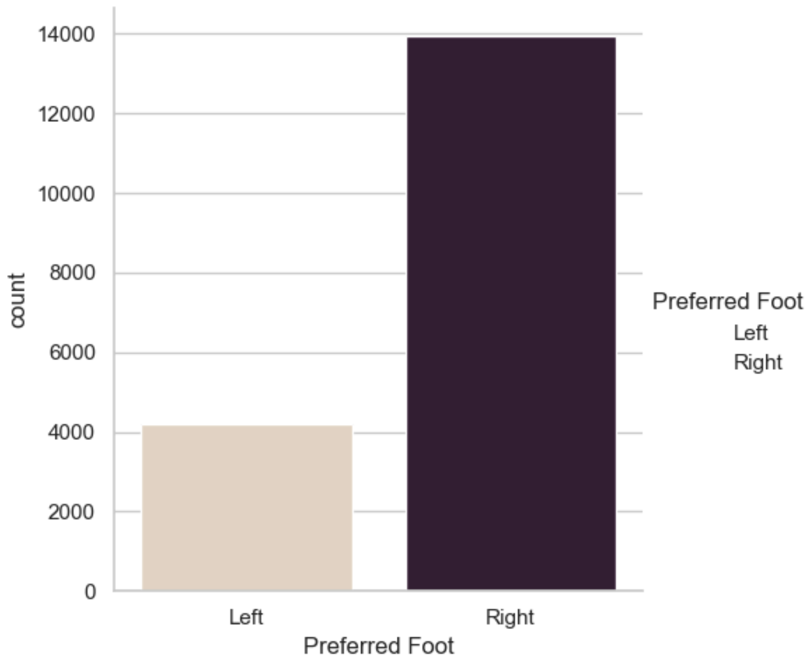


Seaborn `Catplot()` function

- We can use Seaborn `Catplot()` function to plot categorical scatterplots.
- The default representation of the data in `catplot()` uses a scatterplot.
- It helps to draw figure-level interface for drawing categorical plots onto a facetGrid.
- This function provides access to several axes-level functions that show the relationship between a numerical and one or more categorical variables using one of several visual representations.
- The `kind` parameter selects the underlying axes-level function to use.

We can use the kind parameter to draw different plot kin to visualize the same data. We can use the Seaborn `catplot()` function to draw a `countplot()` as follows-

In [32]:
```
g=sns.catplot(x='Preferred Foot',kind='count',palette='ch:.25',data=fifa19)
```



Explore `International Reputation` variable

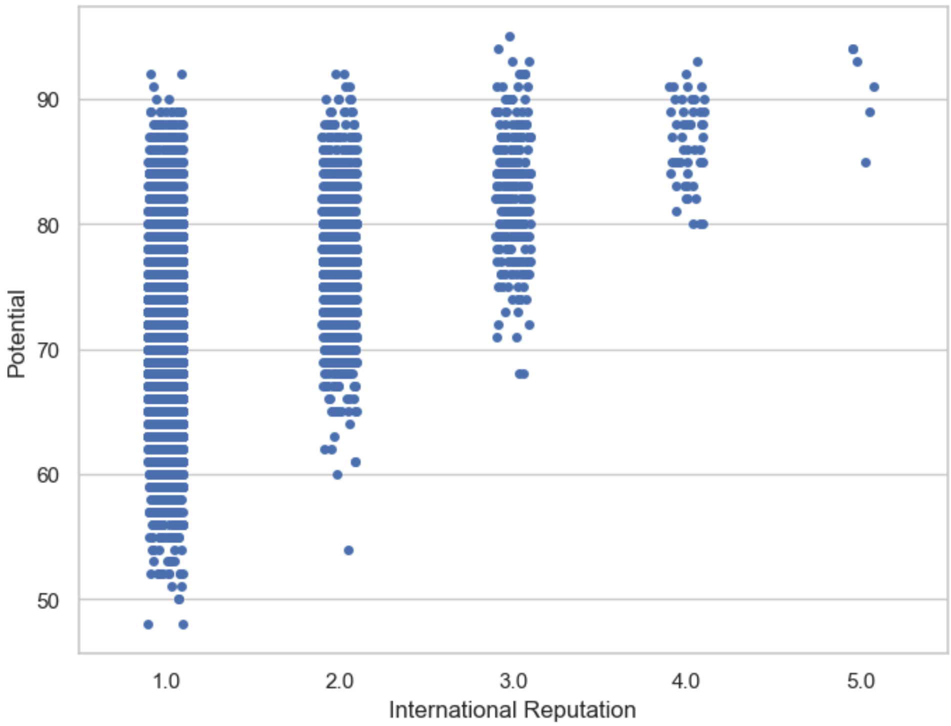
In [34]:
```
# Check the number of unique values in `International Reputation` variable

fifa19['International Reputation'].nunique()
```

Out[34]: 5

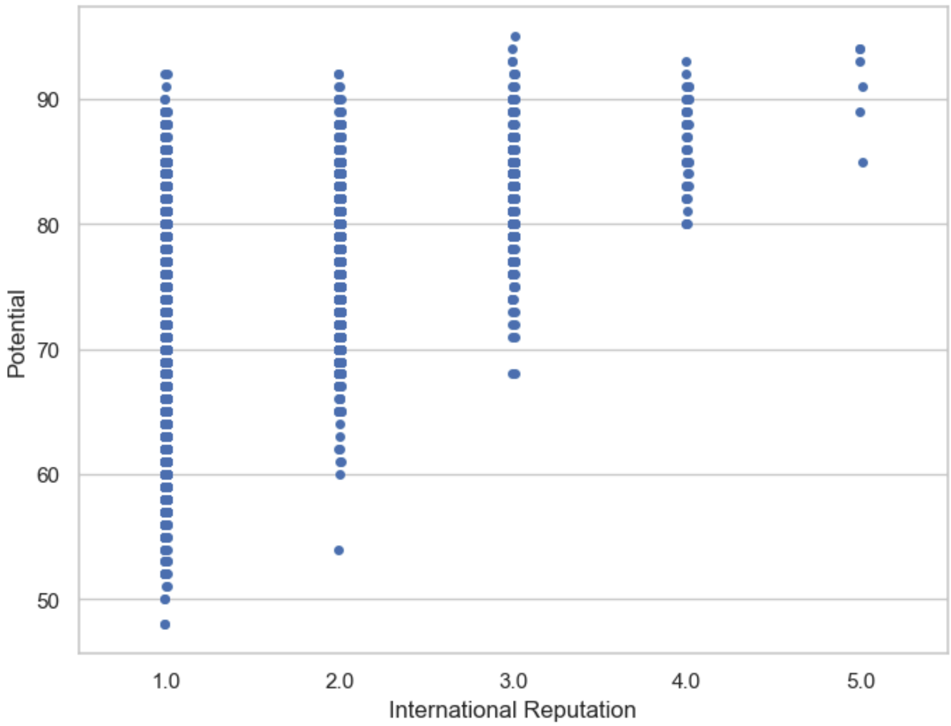
Seaborn `Stripplot()` function

- This function draws a scatterplot where one variable is categorical.
- A strip plot can be drawn on its own, but it is also a good complement to a box or violin plot in cases where we want to show all observations along with some representation of the underlying distribution.
- I will plot a stripplot with `International Reputation` as categorical variable and `Potential` as the other variable.

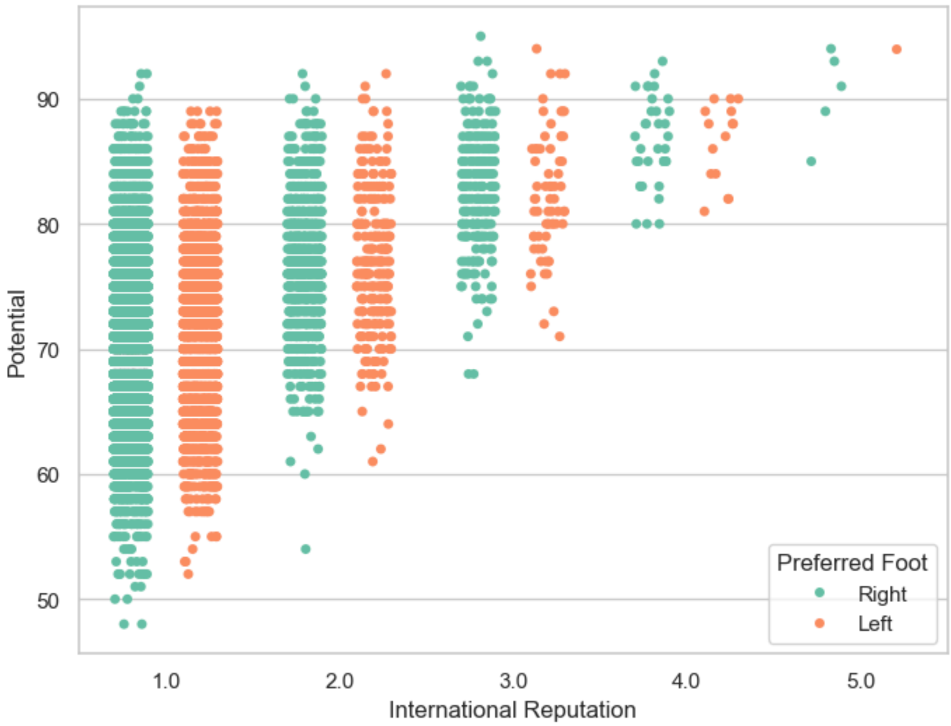
In [36]:
```
f,ax=plt.subplots(figsize=(8,6))
sns.stripplot(x='International Reputation',y='Potential',data=fifa19)
plt.show()
```



In [37]:
```
# we can add jitter to bring out the distribution of values as follows:-

f,ax=plt.subplots(figsize=(8,6))
sns.stripplot(x='International Reputation',y='Potential',data=fifa19,jitter=0.01)
plt.show()
```

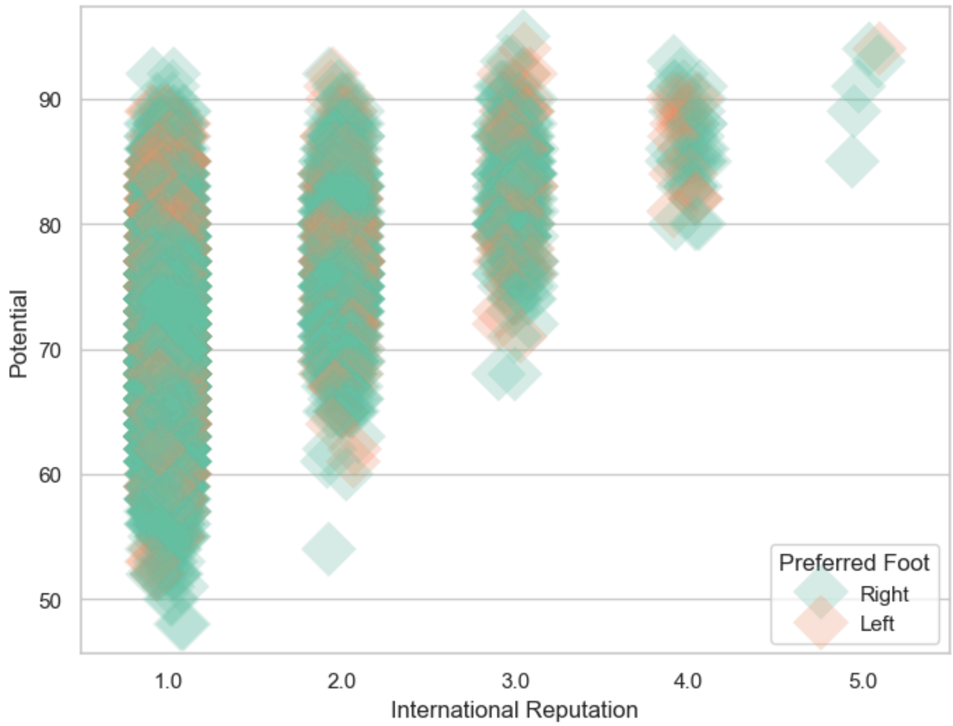


In [38]:
```
# we can nest the strips within a sceond categorical variable 'Preferred Foot' as follow:-

f, ax = plt.subplots(figsize=(8, 6))
sns.stripplot(x="International Reputation", y="Potential", hue="Preferred Foot",
                   data=fifa19, jitter=0.2, palette="Set2", dodge=True)
plt.show()
```



In [39]:

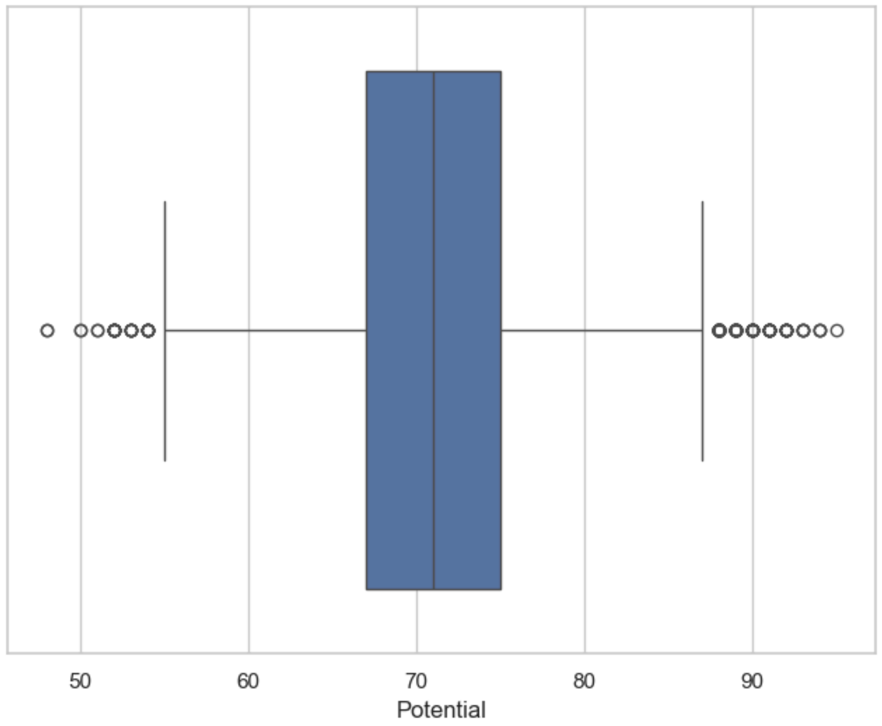
```
# we can draw strips wuth large points and different aeshtetics as follows:-

f,ax=plt.subplots(figsize=(8,6))
sns.stripplot(x='International Reputation',y='Potential',hue='Preferred Foot',
              data=fifa19,palette='Set2',size=20,marker='D',
              edgecolor='gray',alpha=.25)
plt.show()
```



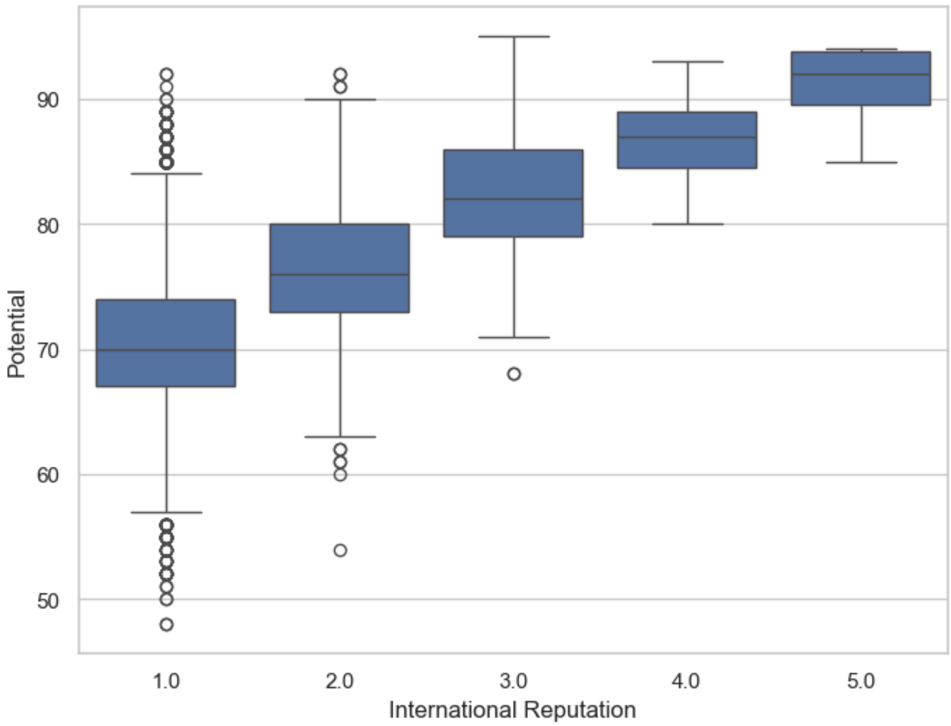
Seaborn `boxplot()` function

- This function draws a box plot to show distributions with respect to categories.
- A box plot (or box-and-whisker plot) shows the distribution of quantitative data in a way that facilitates comparisons between variables or across levels of a categorical variable.
- The box shows the quartiles of the dataset while the whiskers extend to show the rest of the distribution, except for points that are determined to be "outliers" using a method that is a function of the inter-quartile range.
- I will plot the boxplot of the `Potential` variable as follows-

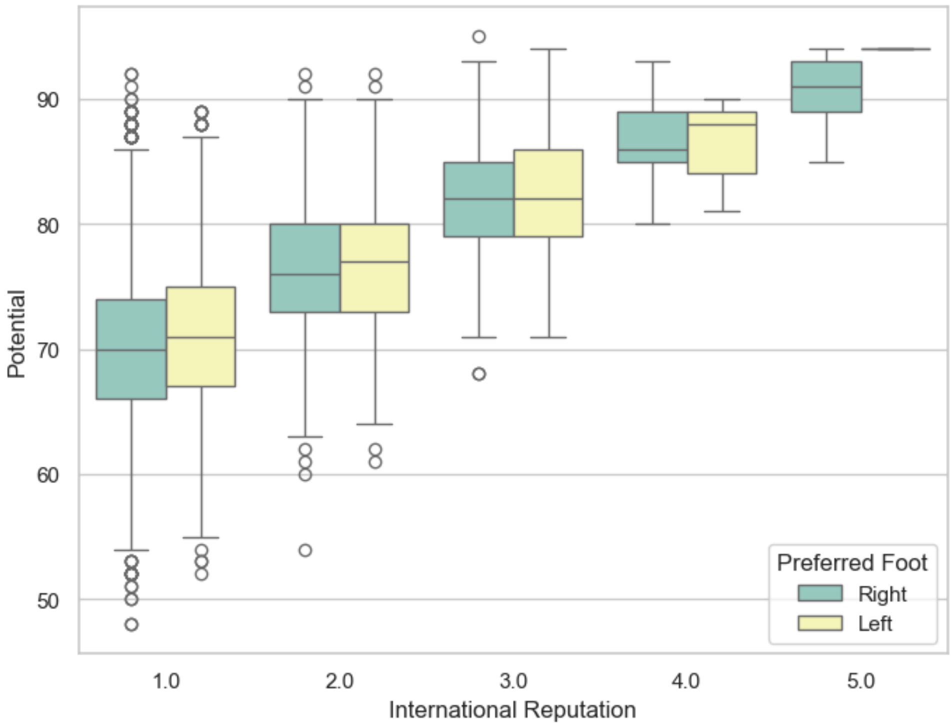
In [41]:
```
f,ax=plt.subplots(figsize=(8,6))
sns.boxplot(x=fifa19['Potential'])
plt.show()
```



In [42]:
```
# we can draw the vertical boxplot gropued by the categorical variable International Reputation as follows:-

f,ax=plt.subplots(figsize=(8,6))
sns.boxplot(x='International Reputation',y='Potential',data=fifa19)
plt.show()
```



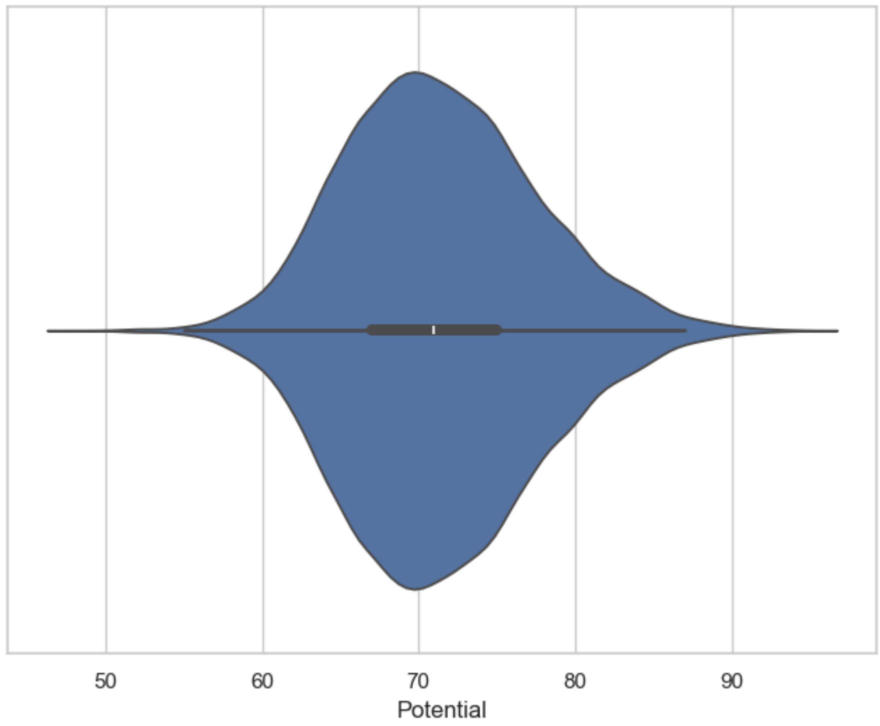
In [43]:
```
# we can draw a box boxplot with nested grouping by two categorical variables as follows:-

f, ax=plt.subplots(figsize=(8,6))
sns.boxplot(x='International Reputation',y='Potential',hue='Preferred Foot',data=fifa19,palette='Set3')
plt.show()
```



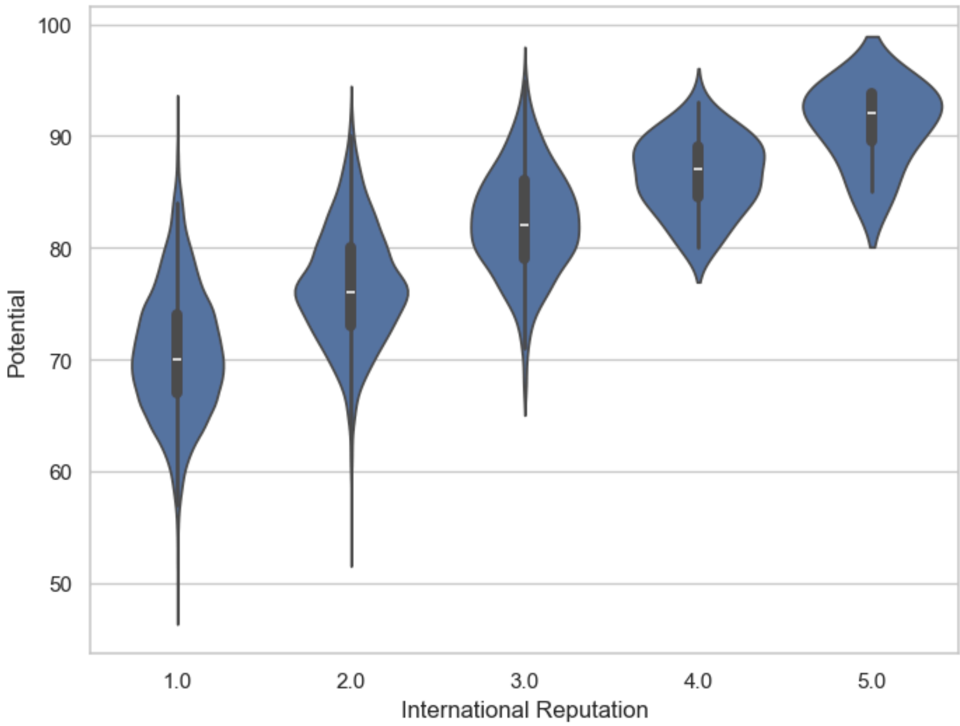
Seaborn `violinplot()` function

- 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.
- I will plot the violinplot of `Potential` variable as follows-

In [45]:
```
f,ax=plt.subplots(figsize=(8,6))
sns.violinplot(x=fifa19['Potential'])
plt.show()
```

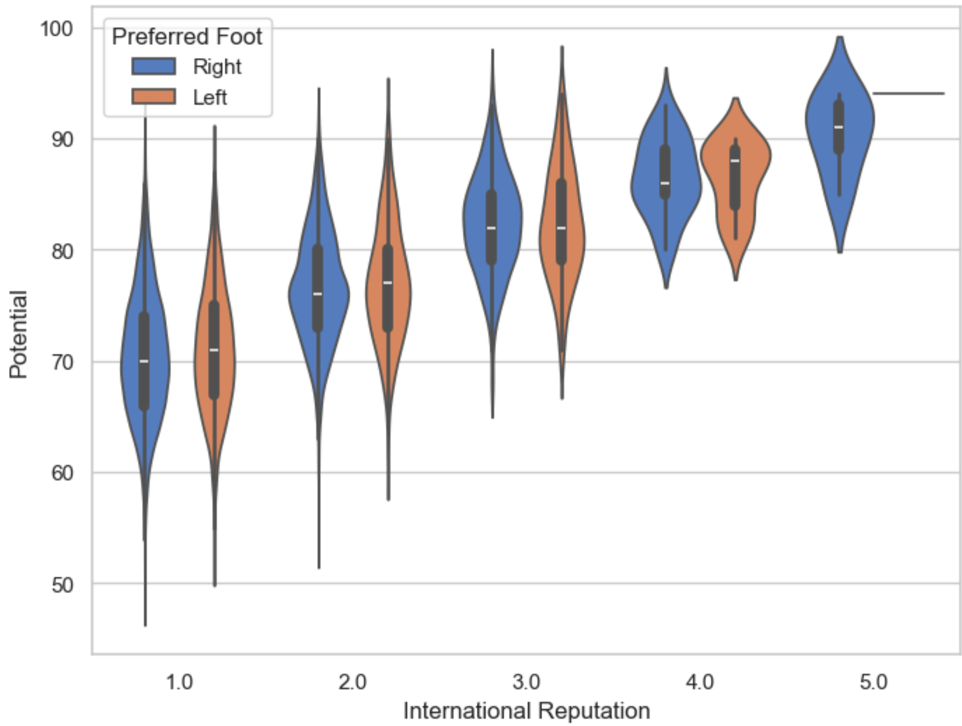


In [46]:
```
# we camn draw the vertical voilinplot  grouped by categorical variable International Repuation as follows:-

f,ax=plt.subplots(figsize=(8,6))
sns.violinplot(x='International Reputation',y='Potential',data=fifa19)
plt.show()
```



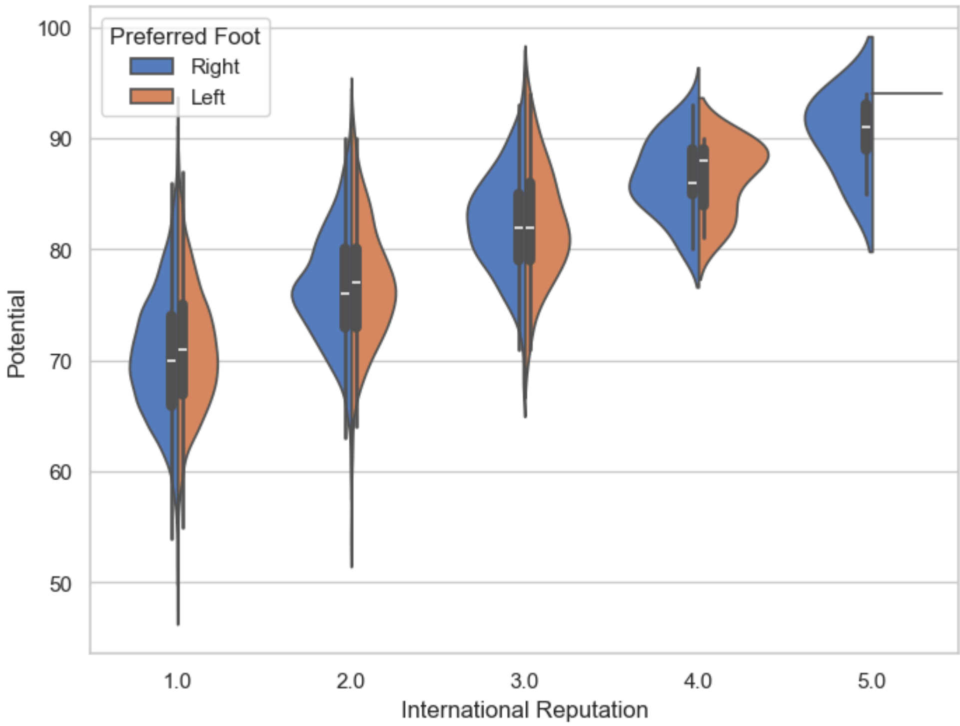
In [47]:

```
# we can draw a violinplot with nested grouping grouping by two categorical varaibles as follows:-

f, ax = plt.subplots(figsize=(8, 6))
sns.violinplot(x="International Reputation", y="Potential", hue="Preferred Foot", data=fifa19, palette="muted")
plt.show()
```



In [48]:

```
# we can draw split violibs to compare the across the hue variable as follows:-

f,ax=plt.subplots(figsize=(8,6))
sns.violinplot(x='International Reputation',y='Potential',hue='Preferred Foot',
               data=fifa19,palette='muted',split=True)
plt.show()
```

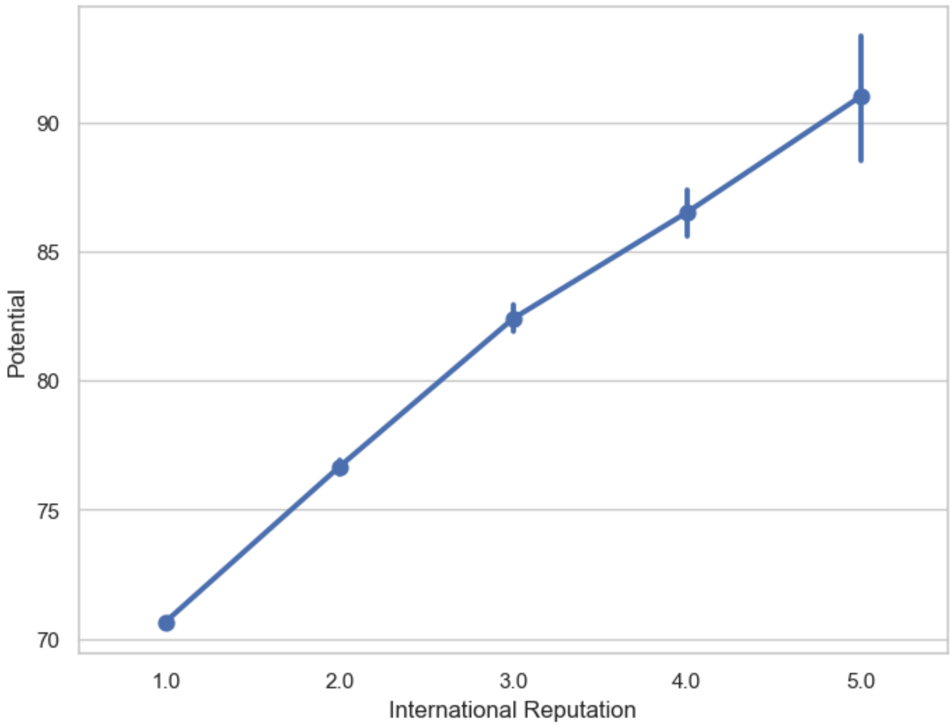


Seaborn `pointplot()` function

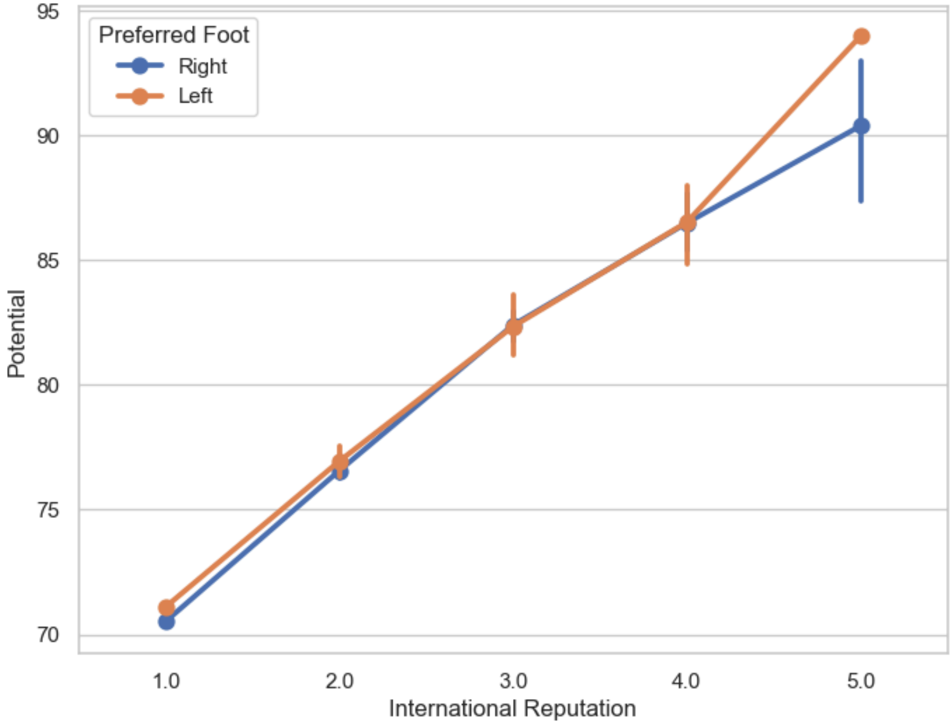
- This function show point estimates and confidence intervals using scatter plot glyphs.
- A point plot represents an estimate of central tendency for a numeric variable by the position of scatter plot points and provides some indication of the uncertainty around that estimate using error bars.

In [50]:

```
f,ax=plt.subplots(figsize=(8,6))
sns.pointplot(x='International Reputation',y='Potential',data=fifa19)
plt.show()
```

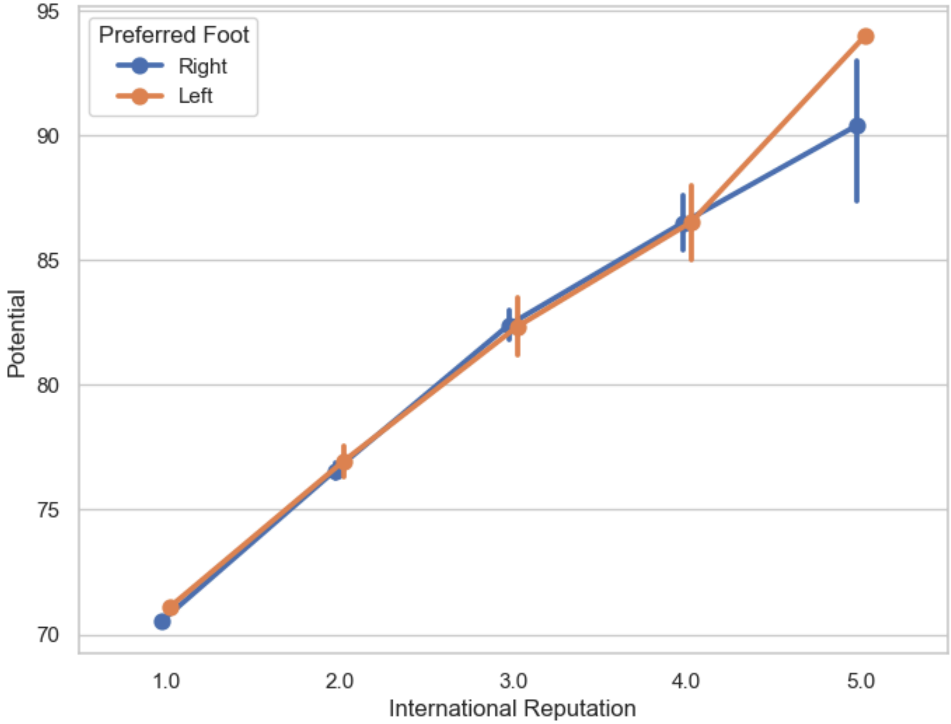


In [51]:
```
# we can draw a set of vertical poinrs wirhe nested grouping by a two variables as folows:-

f,ax=plt.subplots(figsize=(8,6))
sns.pointplot(x='International Reputation',y='Potential',hue='Preferred Foot',data=fifa19)
plt.show()
```

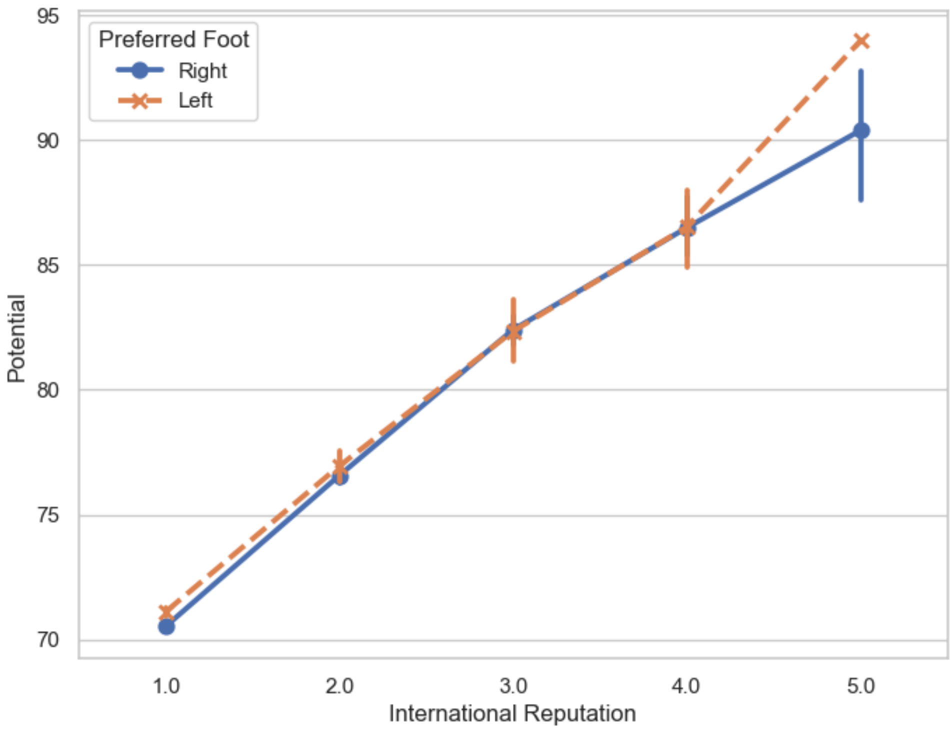


In [52]:

```
# we can separate the points the points different hue levels along the categorical axis as follows:-

f,ax=plt.subplots(figsize=(8,6))
sns.pointplot(x='International Reputation',y='Potential',hue='Preferred Foot',data=fifa19,dodge=True)
plt.show()
```



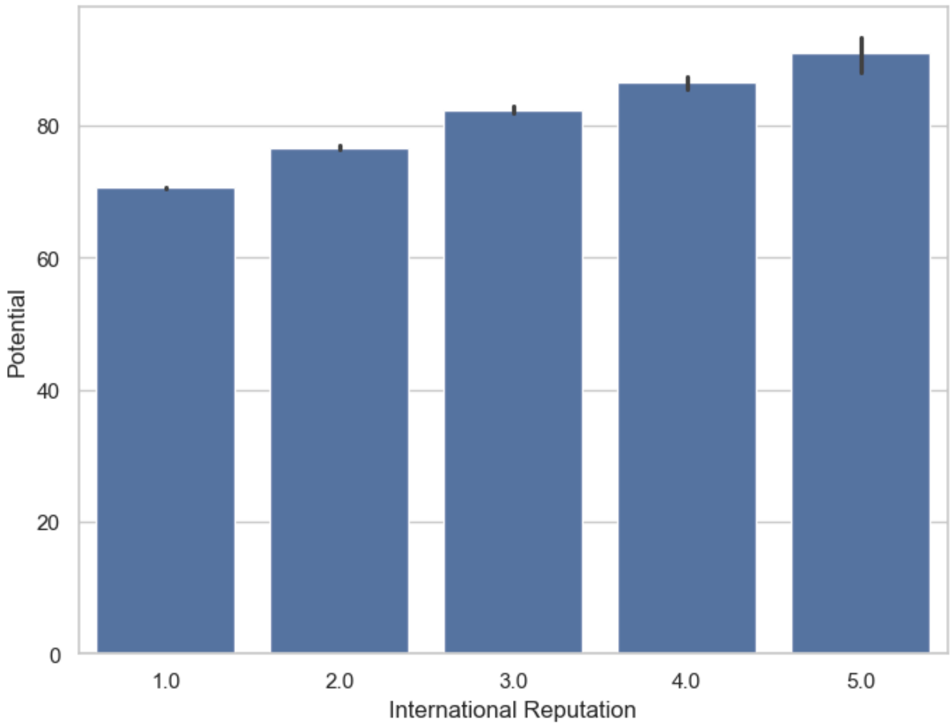
In [53]:
```
# we can use a different marker and line style for the hue levels as follows:-

f,ax=plt.subplots(figsize=(8,6))
sns.pointplot(x='International Reputation',y='Potential',hue='Preferred Foot',
              data=fifa19,markers=['o','x'],linestyles=['-','--'])
plt.show()
```



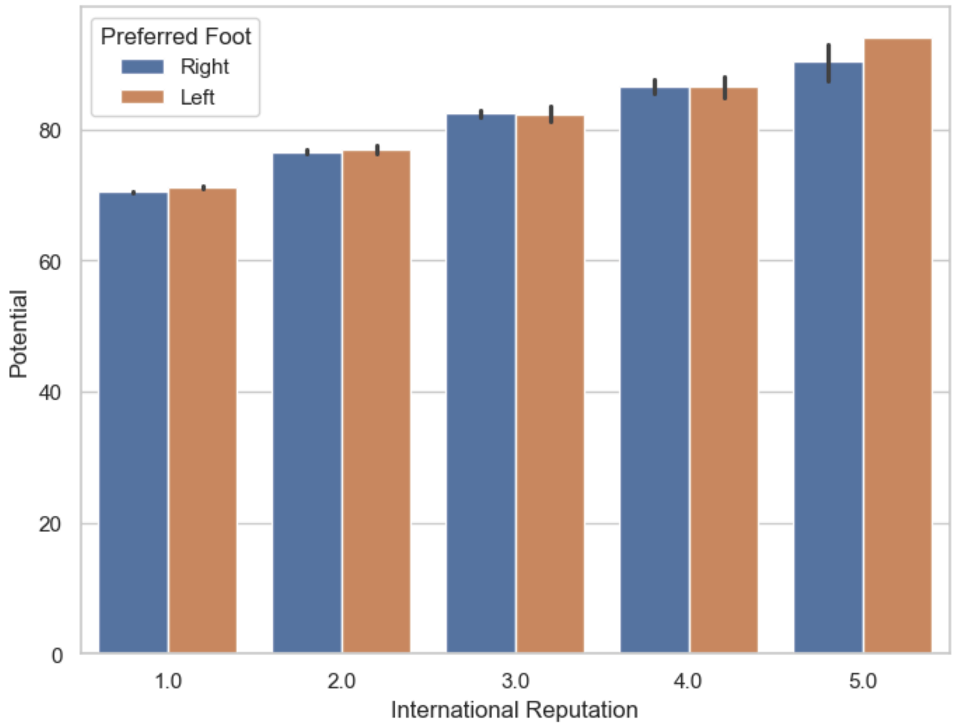
Seaborn `barplot()` function

- This function show point estimates and confidence intervals as rectangular bars.
- A bar plot represents an estimate of central tendency for a numeric variable with the height of each rectangle and provides some indication of the uncertainty around that estimate using error bars.
- Bar plots include 0 in the quantitative axis range, and they are a good choice when 0 is a meaningful value for the quantitative variable, and you want to make comparisons against it.
- We can plot a barplot as follows-

In [55]:
```
f,ax=plt.subplots(figsize=(8,6))
sns.barplot(x='International Reputation',y='Potential',data=fifa19)
plt.show()
```

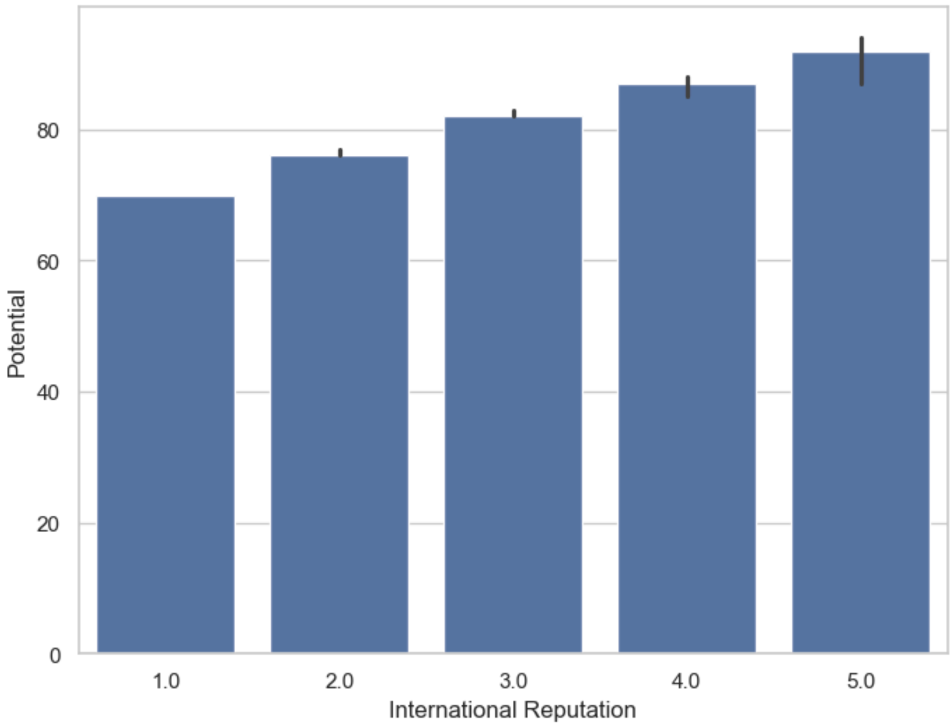


In [56]:
```
# we can drae a set of vertical barplots with nested grouping by a two variables as follows:-

f,ax=plt.subplots(figsize=(8,6))
sns.barplot(x='International Reputation',y='Potential',hue='Preferred Foot',data=fifa19)
plt.show()
```

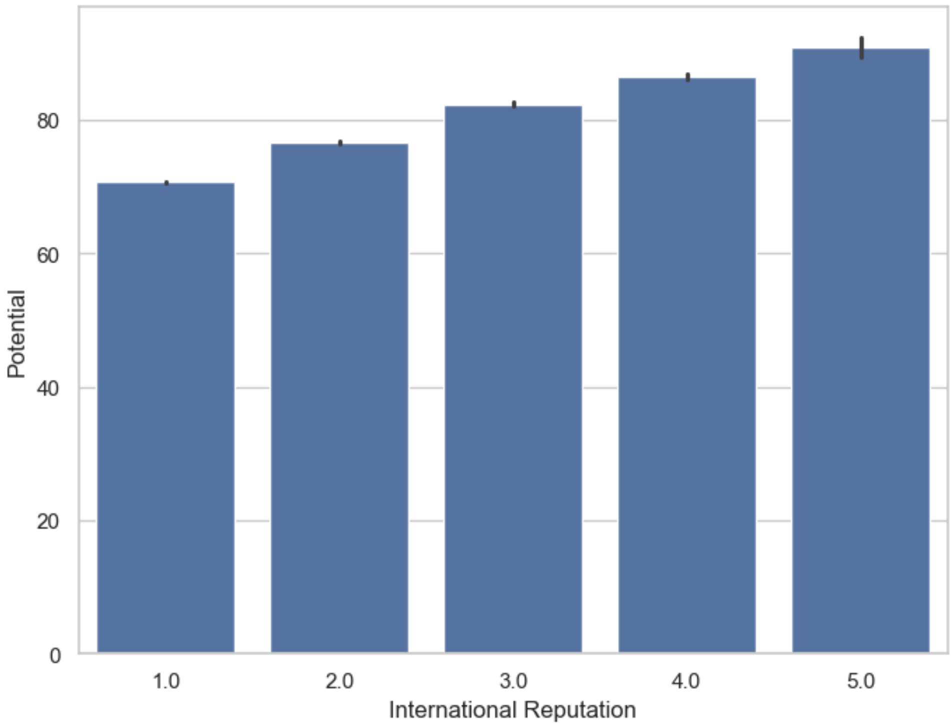


In [57]:

```
# we can use median as the estimate of central tendency as follows:-

from numpy import median
f,ax=plt.subplots(figsize=(8,6))
sns.barplot(x='International Reputation',y='Potential',data=fifa19,estimator=median)
plt.show()
```

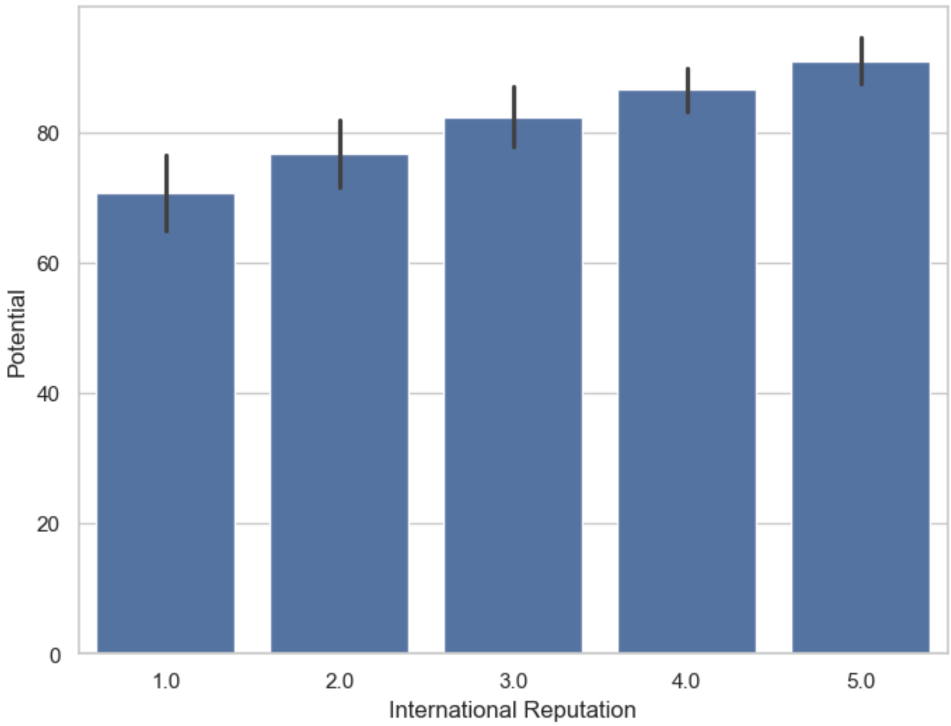


In [58]:
```
# we can show the standard error of the mean with the error bars as follows:-

f,ax=plt.subplots(figsize=(8,6))
sns.barplot(x='International Reputation',y='Potential',data=fifa19,ci=68)
plt.show()
```

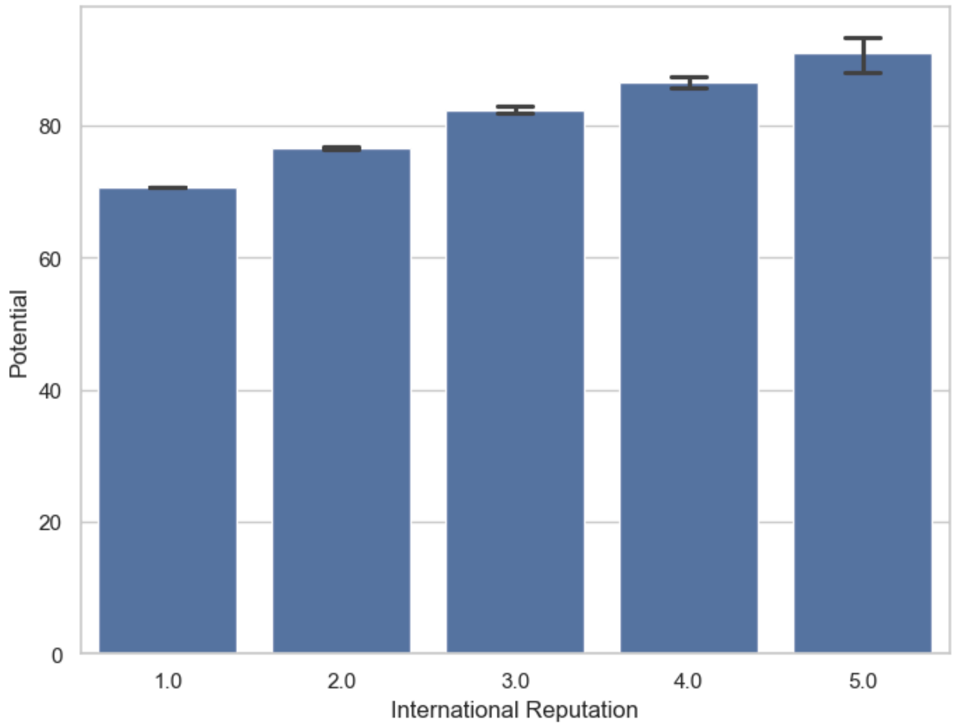


In [59]:

```
# we can show standard dedication of observations instead of a confidence interval as follows:-

f,ax=plt.subplots(figsize=(8,6))
sns.barplot(x='International Reputation',y='Potential',data=fifa19,ci='sd')
plt.show()
```



In [60]:
```
# we can add 'caps' to the error bars as follows:-

f,ax=plt.subplots(figsize=(8,6))
sns.barplot(x='International Reputation',y='Potential',data=fifa19,capsize=0.2)
plt.show()
```

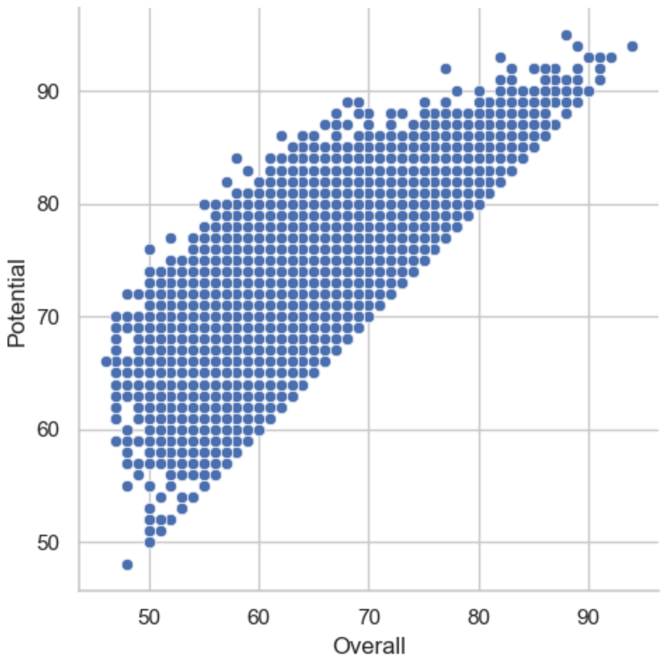


Seaborn `relplot()` function

visualising statistical relationship with seaborn `relpot()` function

- Seaborn `relplot()` function helps us to draw figure-level interface for drawing relational plots onto a FacetGrid.
- This function provides access to several different axes-level functions that show the relationship between two variables with semantic mappings of subsets.
- The `kind` parameter selects the underlying axes-level function to use-
- scatterplot() (with kind="scatter"; the default)
- lineplot() (with kind="line")

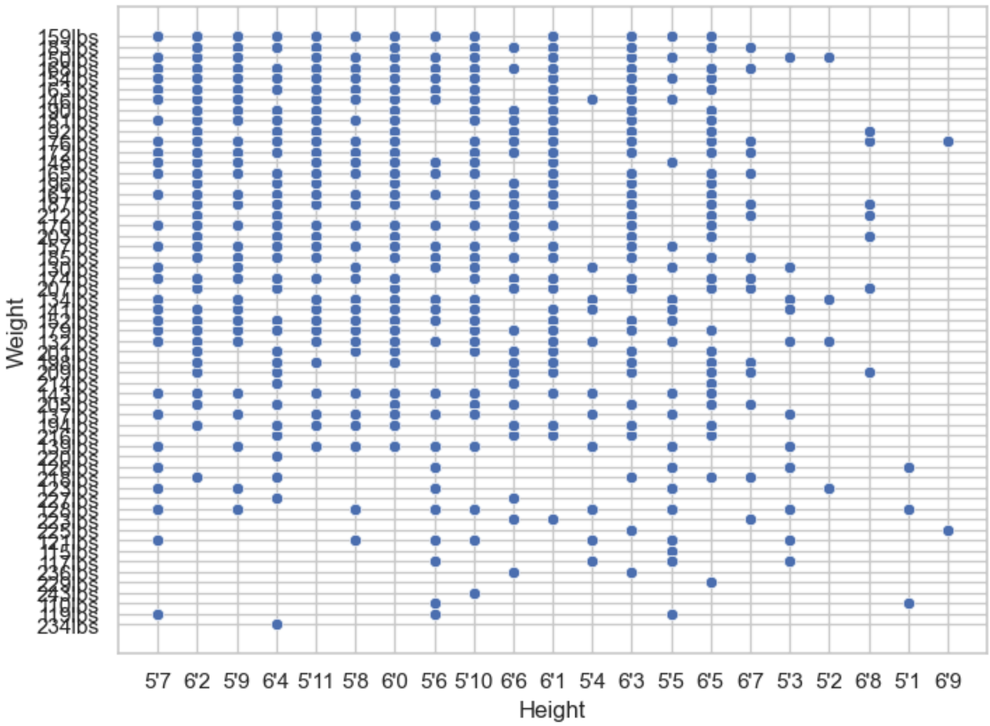
In [62]:
```
g=sns.relplot(x='Overall',y='Potential',data=fifa19)
```



Seaborn `scatterplot()` function

- This function draws a scatter plot with possibility of several semantic groups.
- The relationship between x and y can be shown for different subsets of the data using the `hue` , `size` and `style` parameters.
- These parameters control what visual semantics are used to identify the different subsets.

In [64]:
```
f,ax=plt.subplots(figsize=(8,6))
sns.scatterplot(x='Height',y='Weight',data=fifa19)
plt.show()
```

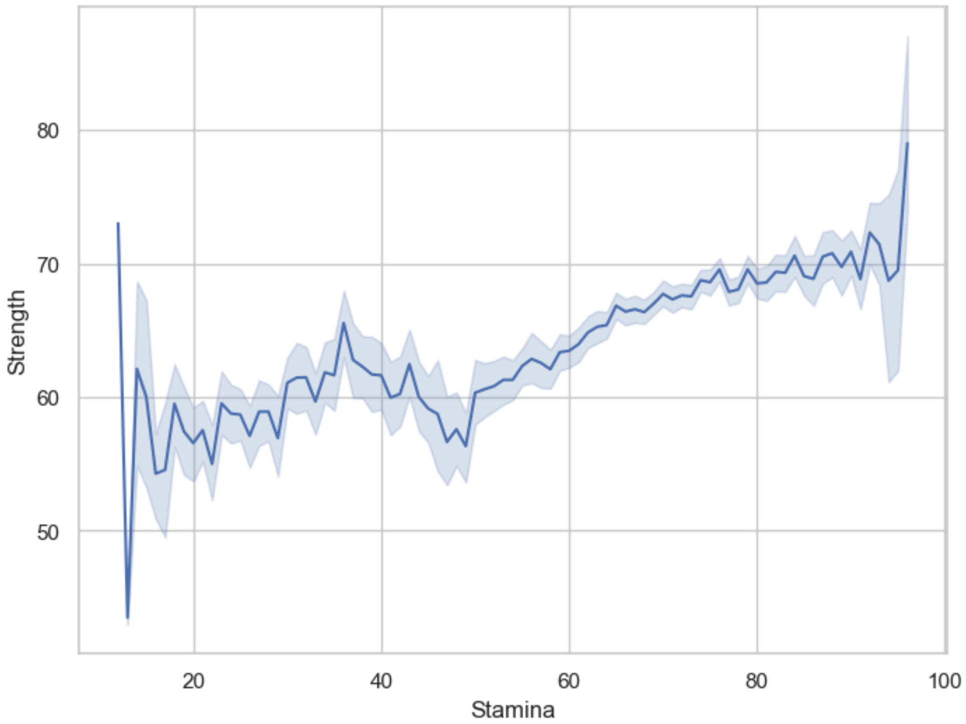


Seaborn `lineplot()` function

- THis function draws a line plot with possibility of several semantic groupings.
- The relationship between x and y can be shown for different subsets of the data using the `hue`, `size` and `style` parameters.
- These parameters control what visual semantics are used to identify the different subsets.

In [66]:

```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.lineplot(x='Stamina',y='Strength',data=fifa19)
plt.show()
```

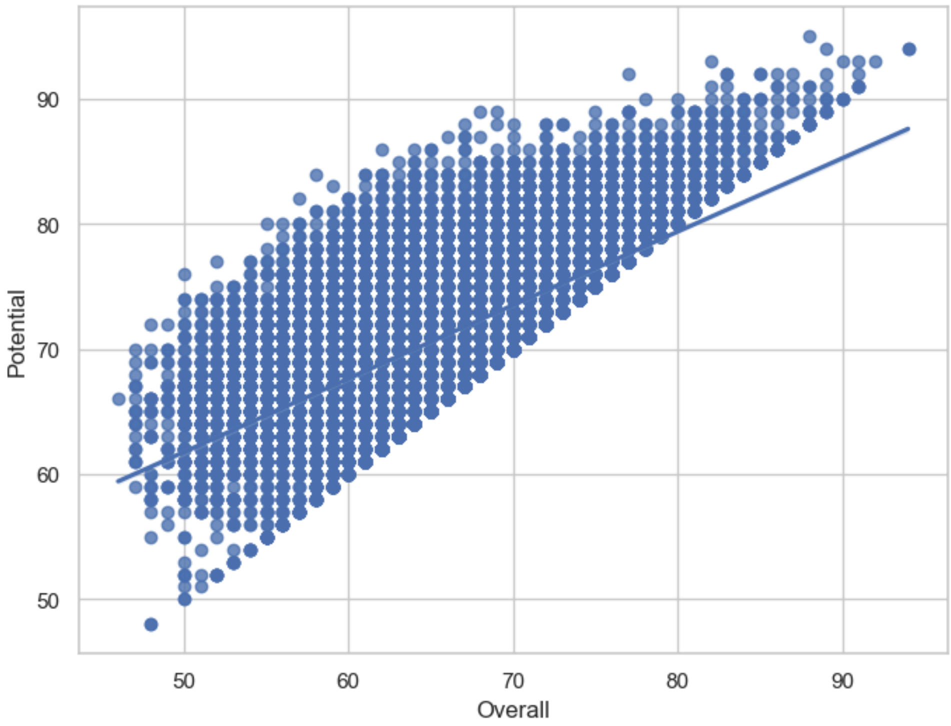


Visualize linear relationship with Seaborn `regplot()` function

Seaborn `regplot()` function

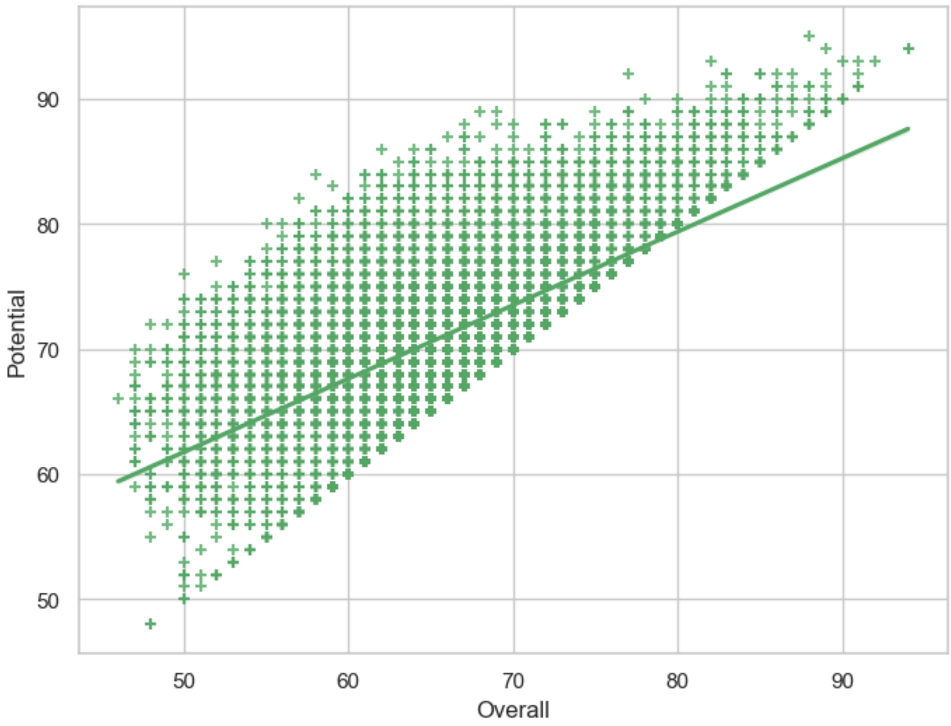
- This function plots data and a linear regression model fit.
- We can plot a linear regression model between `Overall` and `Potential` variable with `regplot()` function as follows-

In [68]:
```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.regplot(x='Overall',y='Potential',data=fifa19)
plt.show()
```

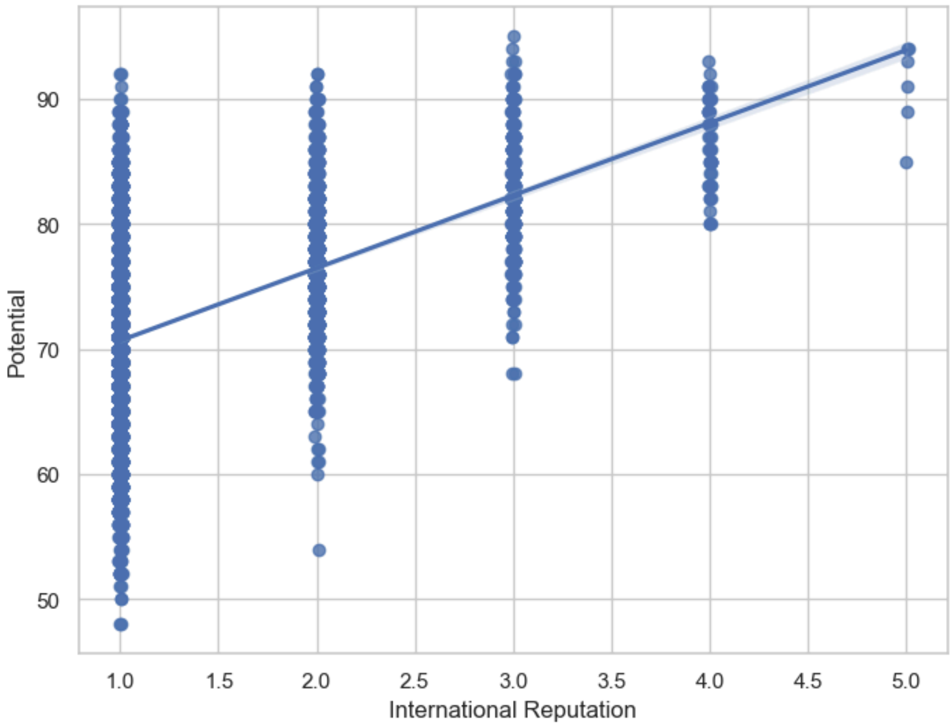


In [69]:

```
# we can use a different color and marker as follows:-

f,ax=plt.subplots(figsize=(8,6))
ax=sns.regplot(x='Overall',y='Potential',data=fifa19,color='g',marker='+')
plt.show()
```



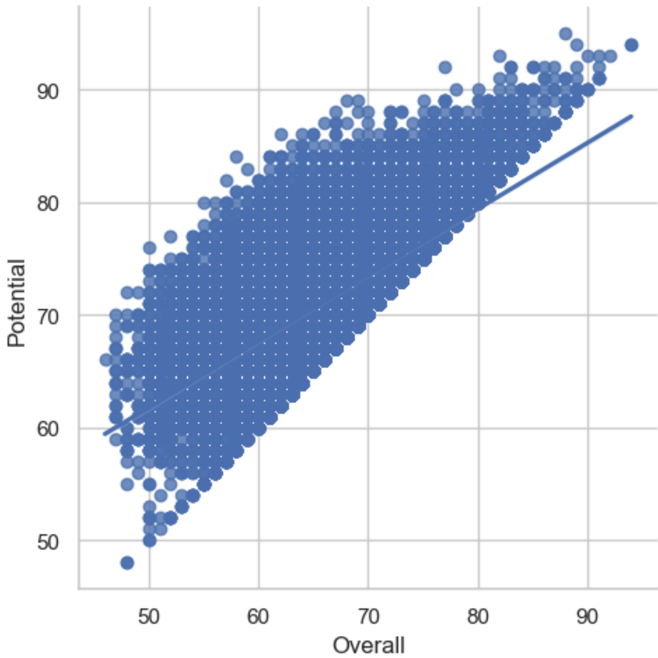
In [70]:
```
# we can plot with a discrete variable and add some jitter as follows:-

f,ax=plt.subplots(figsize=(8,6))
sns.regplot(x='International Reputation',y='Potential',data=fifa19,x_jitter=.01)
plt.show()
```



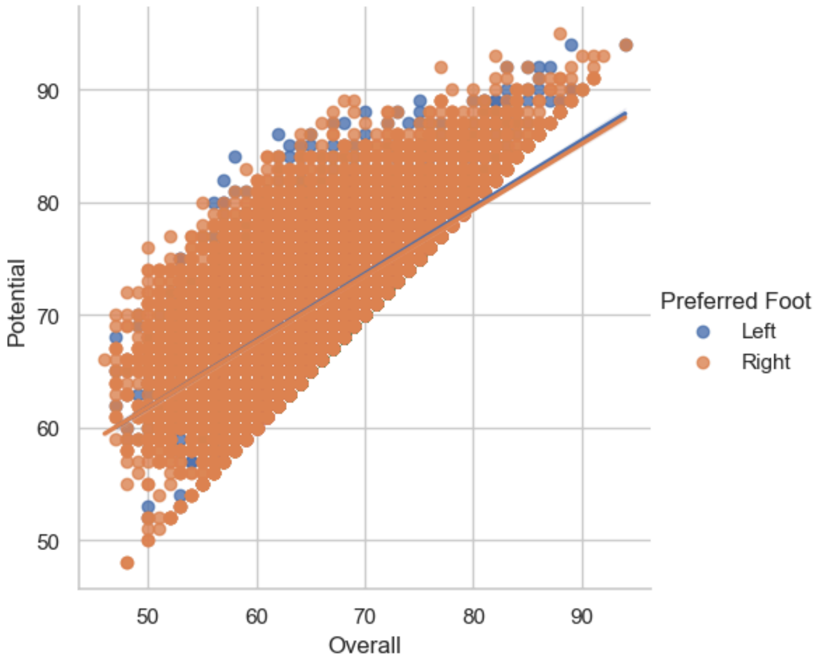
Seaborn `lmplot()` function

- This function plots data and regression model fits across a FacetGrid.
- This function combines `regplot()` and `FacetGrid`.
- It is intended as a convenient interface to fit regression models across conditional subsets of a dataset.
- We can plot a linear regression model between `Overall` and `Potential` variable with `lmplot()` function as follows-

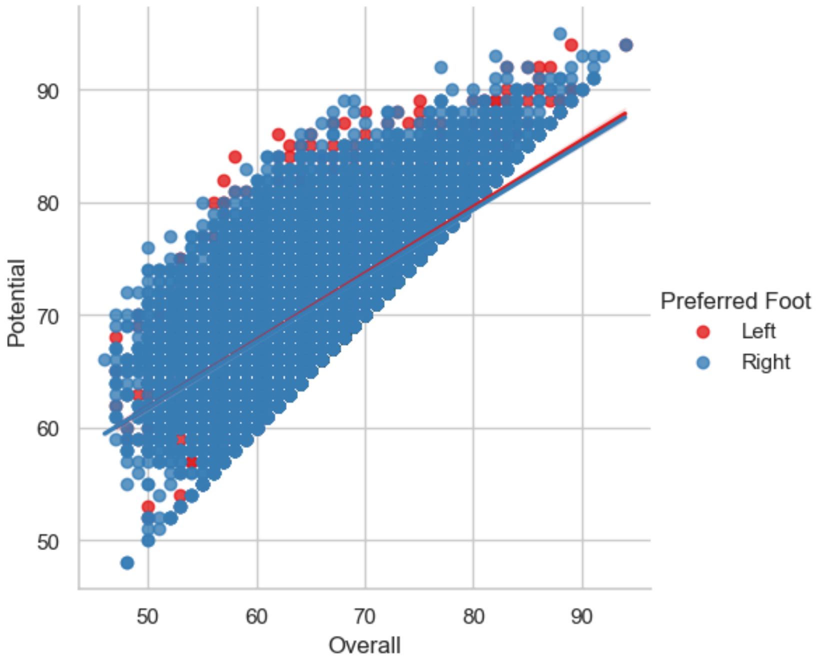
In [72]:
```
g=sns.lmplot(x='Overall',y='Potential',data=fifa19)
```



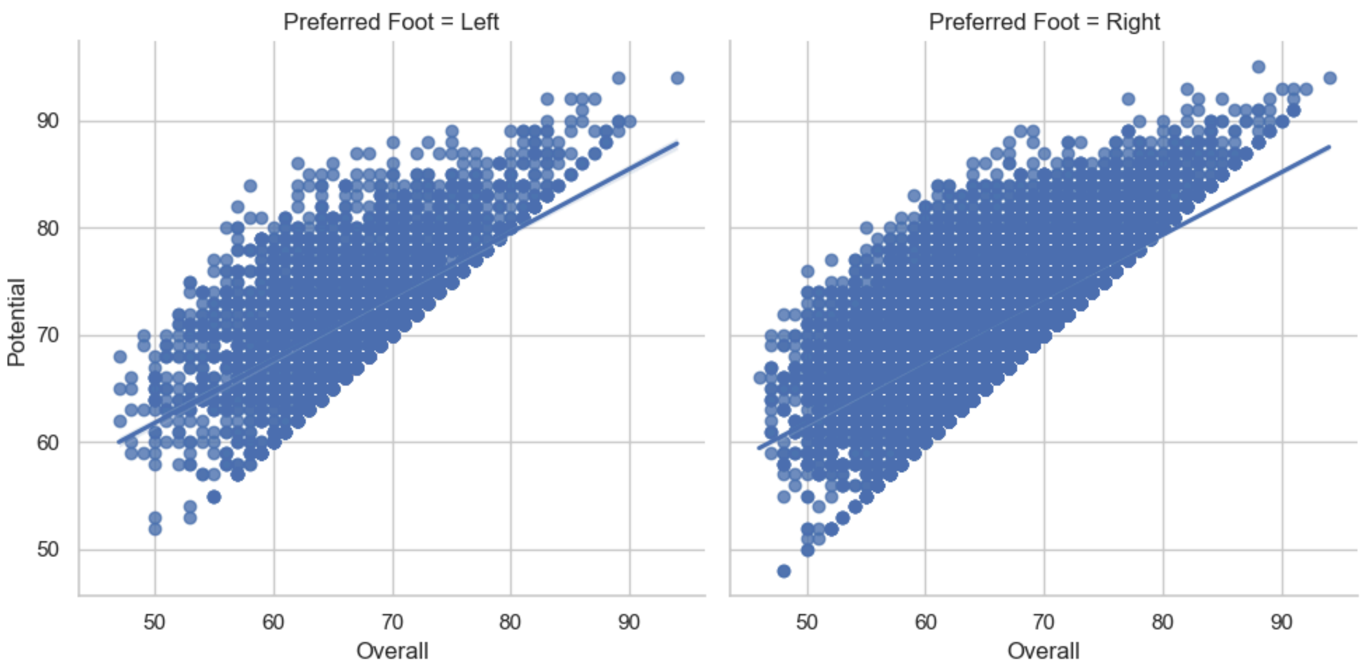
In [73]:
```
# we can condition on a third variable and plot the levels in different colors as follows:-

g=sns.lmplot(x='Overall',y='Potential',hue='Preferred Foot',data=fifa19)
```



In [74]:
```
# we can use a different volor palette as follows:-

g=sns.lmplot(x='Overall',y='Potential',hue='Preferred Foot',data=fifa19,palette='Set1')
```



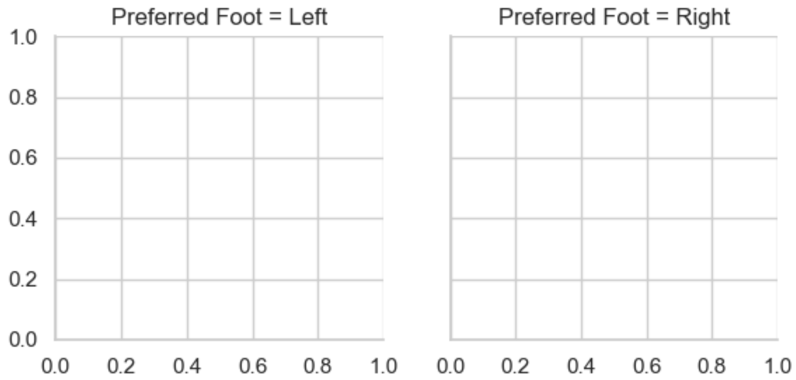
In [75]:
```
# we can plot the levels of the third variable across different columns as follows:-

g=sns.lmplot(x='Overall',y='Potential',col='Preferred Foot',data=fifa19)
```



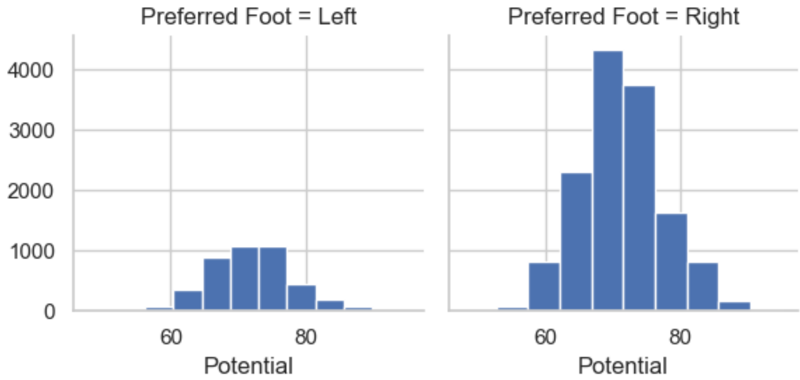
Seaborn `FacetGrid()` function

- The FacetGrid class is useful when you want to visualize the distribution of a variable or the relationship between multiple variables separately within subsets of your dataset.
- A FacetGrid can be drawn with up to three dimensions - `row`, `col` and `hue`. The first two have obvious correspondence with the resulting array of axes - the `hue` variable is a third dimension along a depth axis, where different levels are plotted with different colors.
- The class is used by initializing a FacetGrid object with a dataframe and the names of the variables that will form the `row`, `column` or `hue` dimensions of the grid.
- These variables should be categorical or discrete, and then the data at each level of the variable will be used for a facet along that axis.

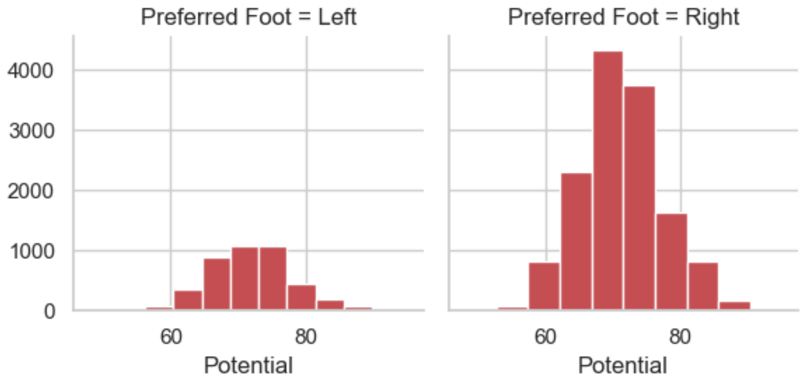
In [77]:
```
# we can initialize 1x2 grid of facets using the fifa19 dataset

g=sns.FacetGrid(fifa19,col='Preferred Foot')
```



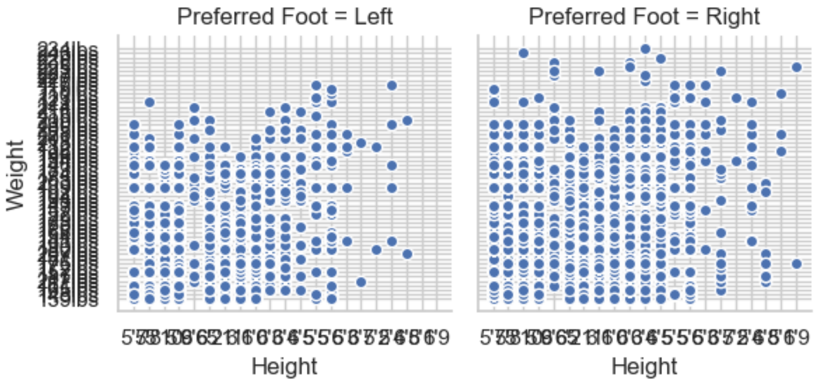
In [78]:
```
# we can draw a univariate plot of 'Potential' variable on each facet as follows:-

g=sns.FacetGrid(fifa19,col='Preferred Foot')
g=g.map(plt.hist,'Potential')
```



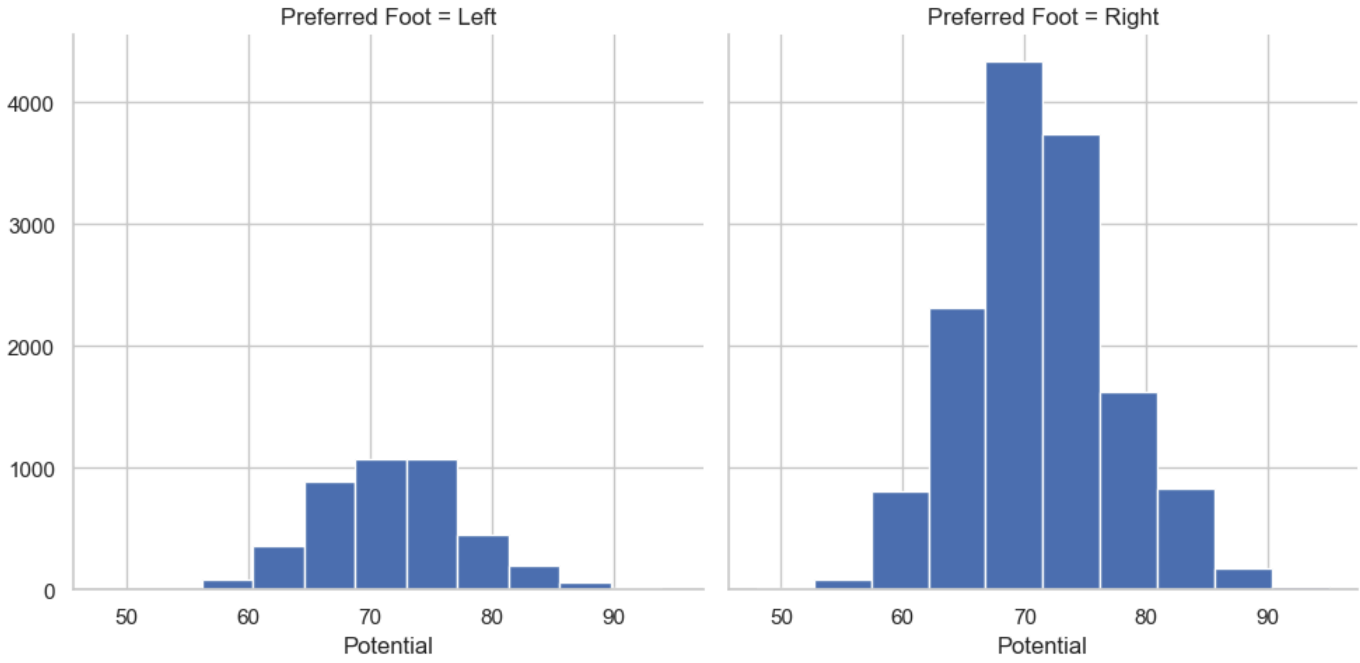
In [79]:
```
g=sns.FacetGrid(fifa19,col='Preferred Foot')
g=g.map(plt.hist,'Potential',bins=10,color='r')
```



In [80]:
```python
# we can plot a bivariate function on each facet as follows:-

g=sns.FacetGrid(fifa19,col='Preferred Foot')
g=(g.map(plt.scatter,'Height','Weight',edgecolor='w').add_legend())
```



In [81]:
```python
# The size of the figure is set by providing the height of each Facet, along with aspects ratio:-

g=sns.FacetGrid(fifa19,col='Preferred Foot',height=5,aspect=1)
g=g.map(plt.hist,'Potential')
```

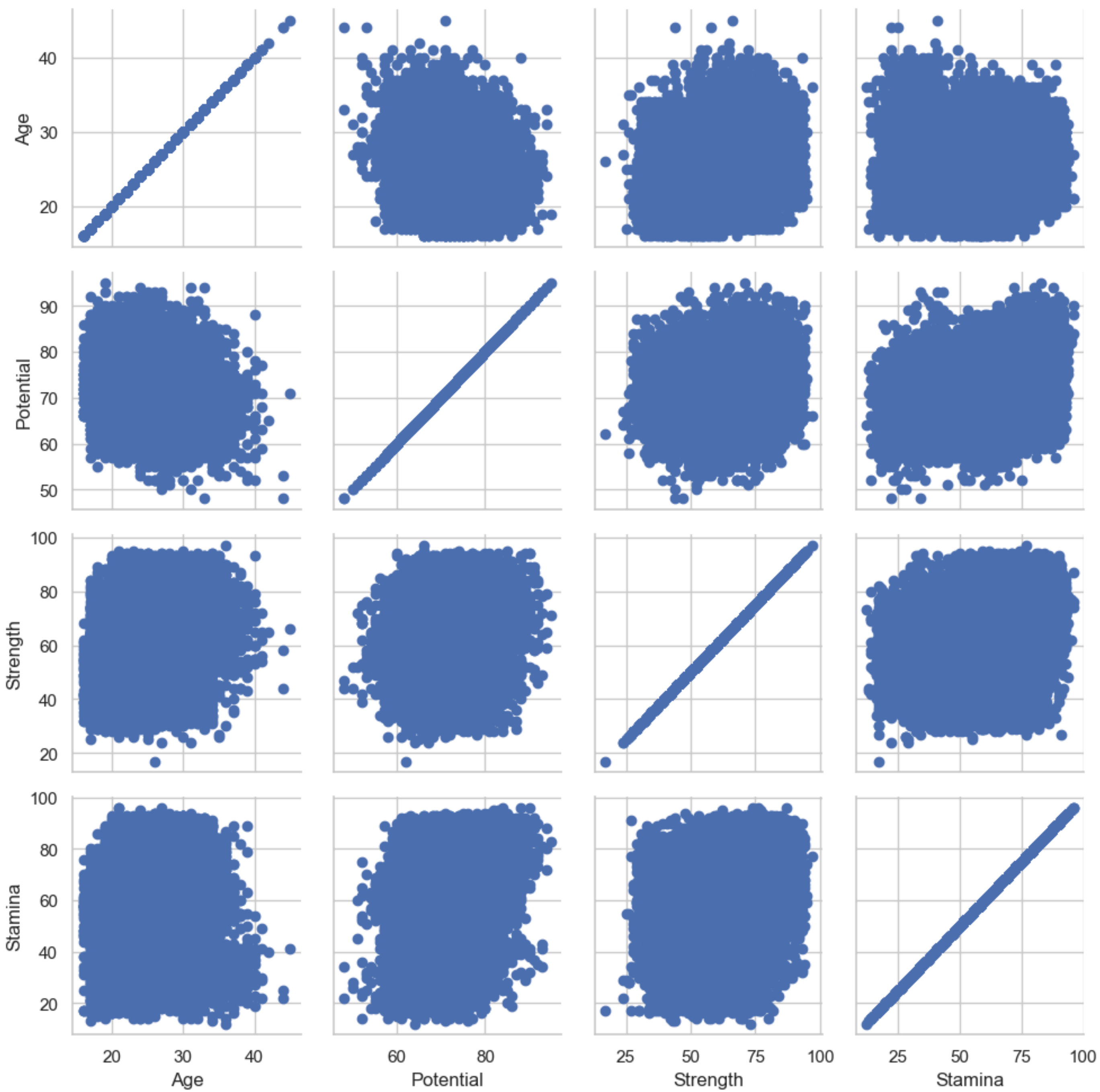


Seaborn `Pairgrid()` function

- This function plots subplot grid for plotting pairwise relationships in a dataset.
- This class maps each variable in a dataset onto a column and row in a grid of multiple axes.
- Different axes-level plotting functions can be used to draw bivariate plots in the upper and lower triangles, and the the marginal distribution of each variable can be shown on the diagonal.
- It can also represent an additional level of conditionalization with the hue parameter, which plots different subets of data in different colors.
- This uses color to resolve elements on a third dimension, but only draws subsets on top of each other and will not tailor the hue parameter for the specific visualization the way that axes-level functions that accept hue will.

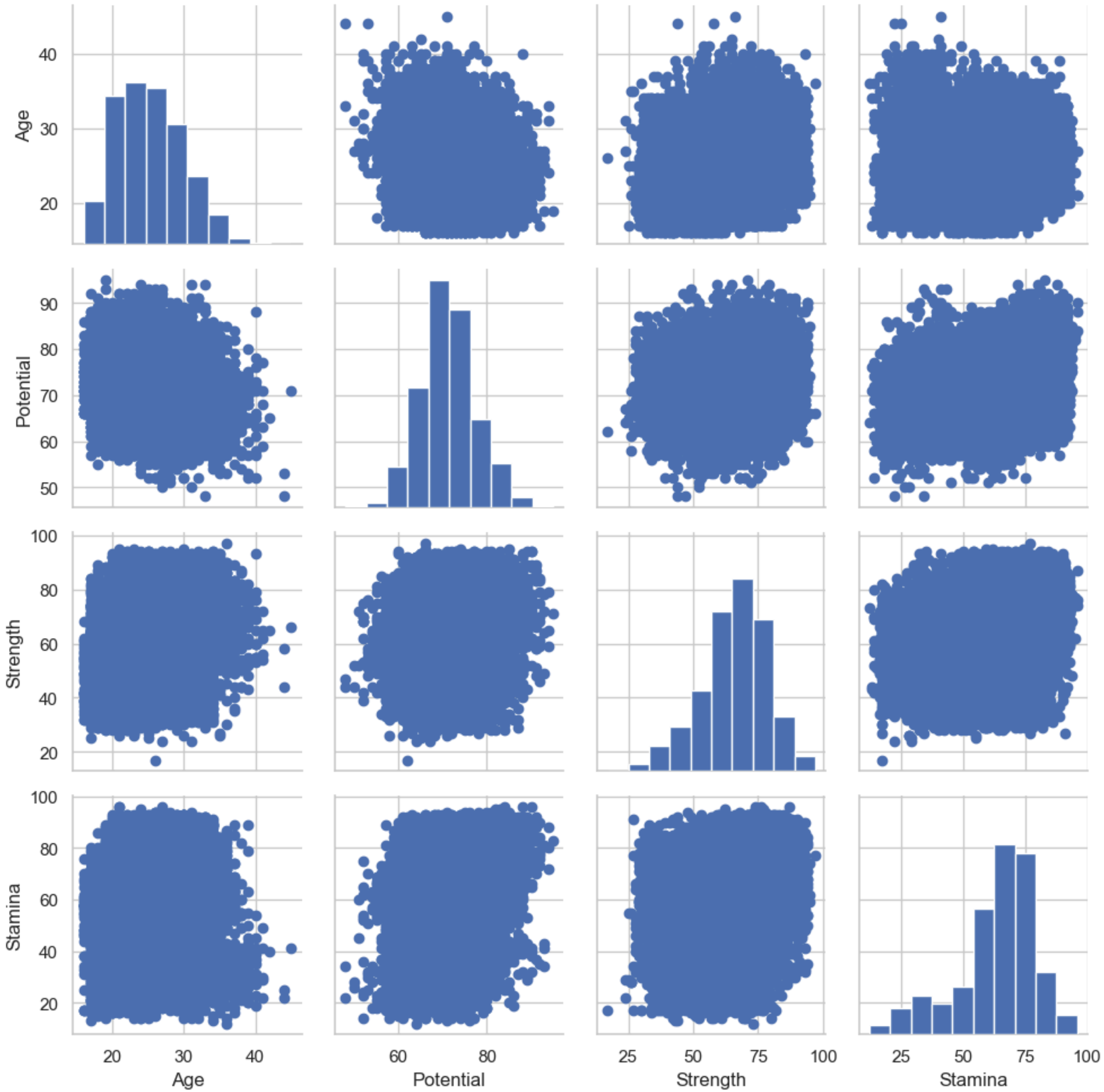
In [83]:
```
fifa19_new=fifa19[['Age','Potential','Strength','Stamina','Preferred Foot']]
```

In [84]:
```
g=sns.PairGrid(fifa19_new)
g=g.map(plt.scatter)
```


Age
Potential
Strength
Stamina
20
30
40
60
80
25
50
75
100
90
70
50

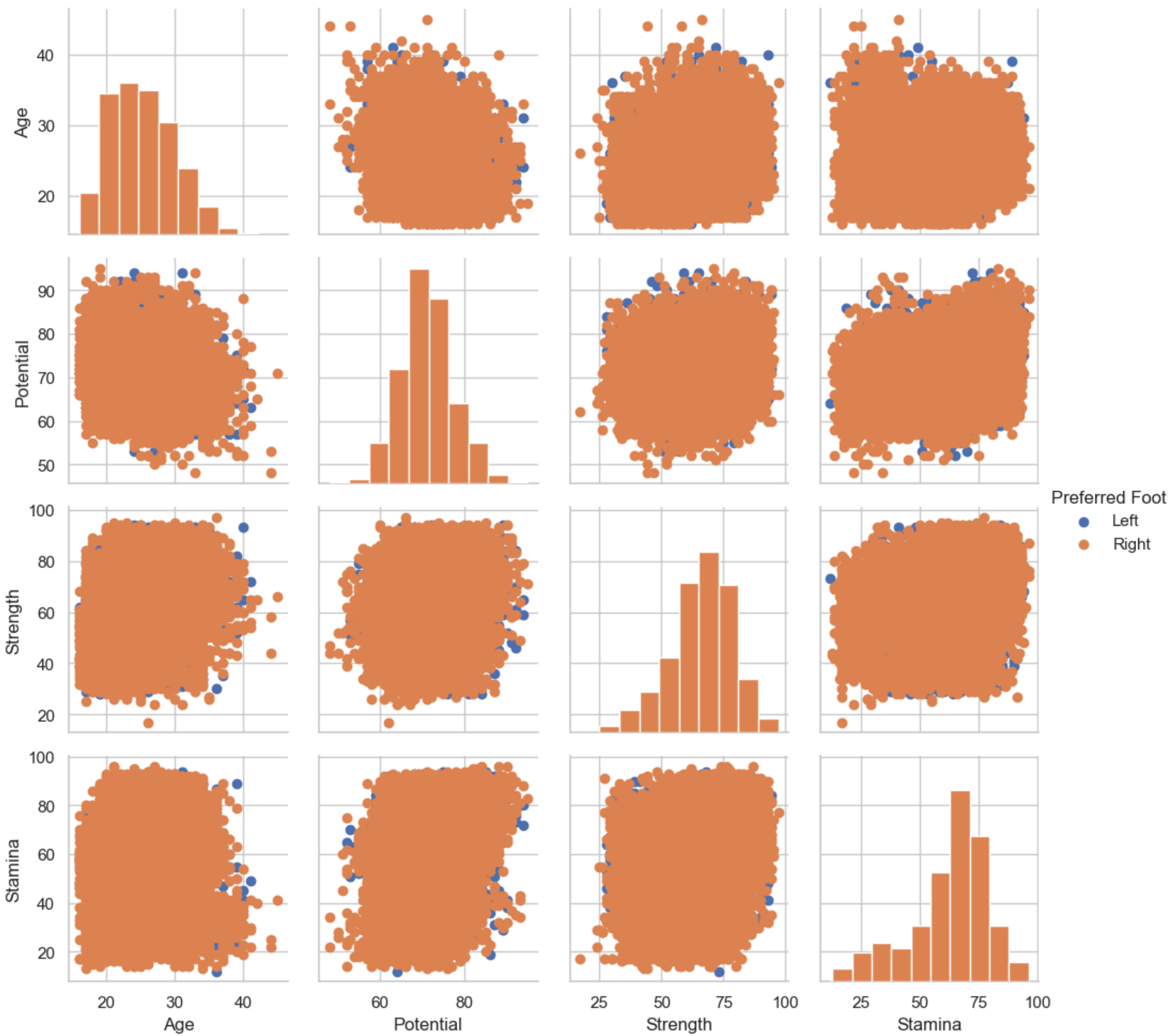
In [85]:

```
# we can show a univriate distribution on the diagonal as follows:-

g=sns.PairGrid(fifa19_new)
g=g.map_diag(plt.hist)
g=g.map_offdiag(plt.scatter)
```

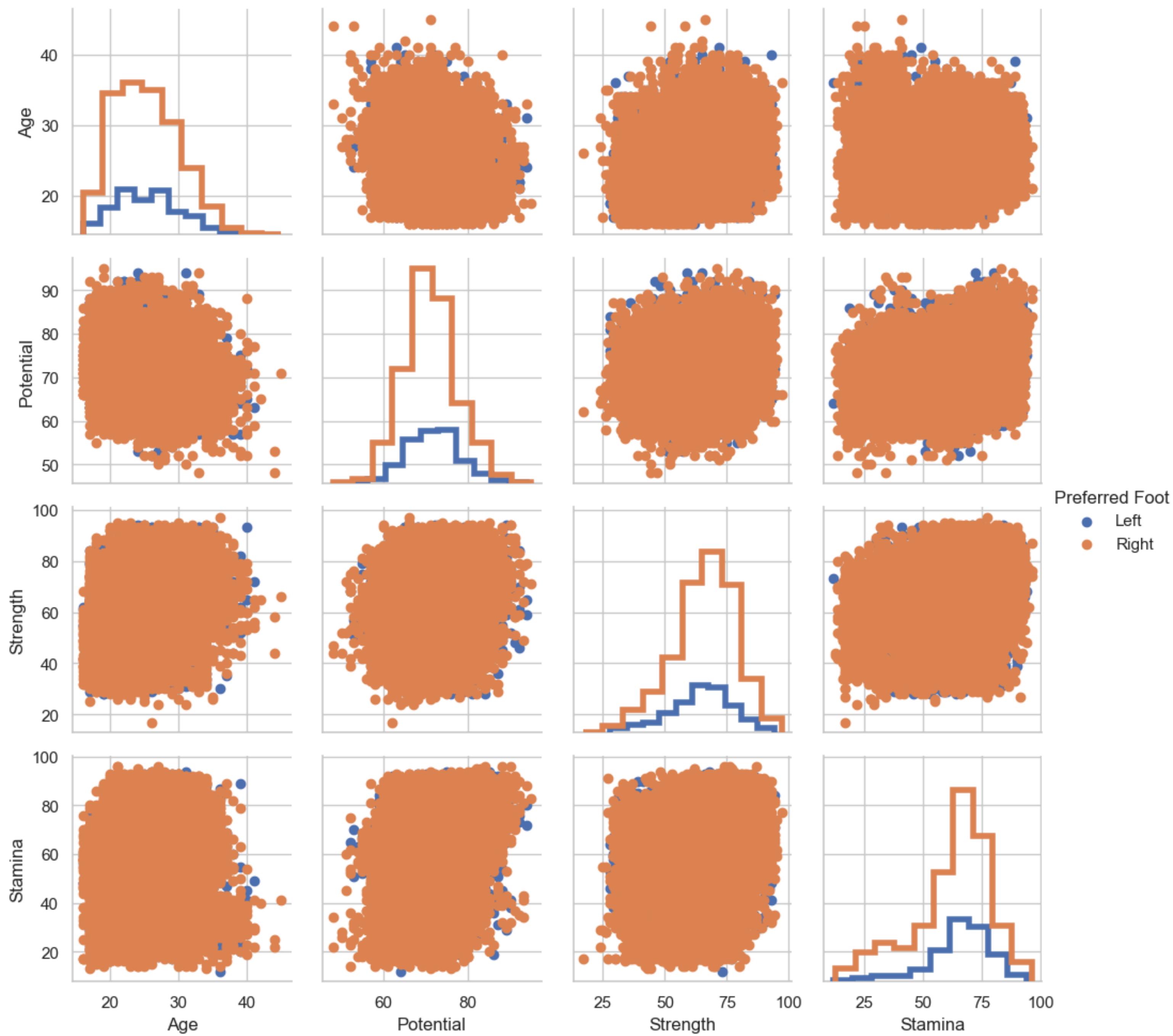

Age
Potential
Strength
Stamina
40
30
20
90
80
70
60
50
100
80
60
40
20
100
80
60
40
20
20
30
40
60
80
25
50
75
100
25
50
75
100
Age
Potential
Strength
Stamina

In [86]:

```
# we can color the points using the categorical variable 'Preferred Foot' as follows:-

g=sns.PairGrid(fifa19_new,hue='Preferred Foot')
g=g.map_diag(plt.hist)
g=g.map_offdiag(plt.scatter)
g=g.add_legend()
```

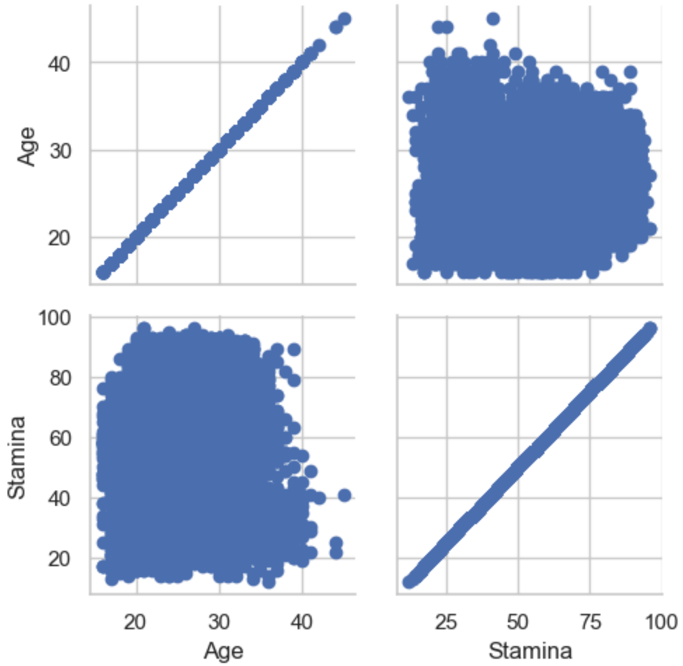

Age
Potential
Strength
Stamina
Preferred Foot
Left
Right

In [87]:
```
# we can use a different style to show multiple histograms as follows:-

g=sns.PairGrid(fifa19_new,hue='Preferred Foot')
g=g.map_diag(plt.hist,histtype='step',linewidth=4)
g=g.map_offdiag(plt.scatter)
g=g.add_legend()
```

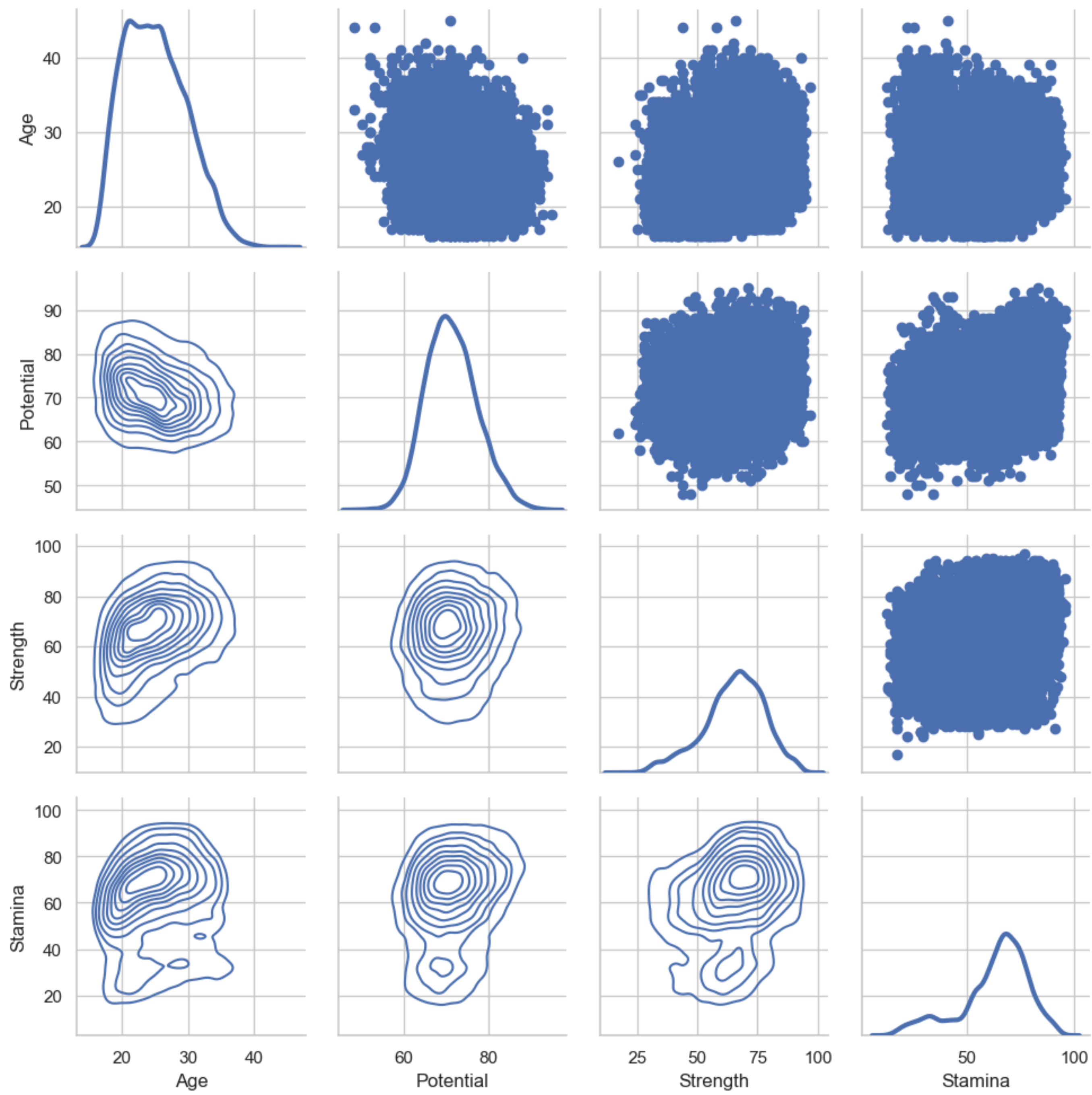


Age
Potential
Strength
Stamina
Preferred Foot
Left
Right

In [88]:
```
# we can plot a subsets of variables as follows:-

g=sns.PairGrid(fifa19_new,vars=['Age','Stamina'])
g=g.map(plt.scatter)
```



In [89]:
```
# we can use different functions on the upper lower traiangles as follows:-

g=sns.PairGrid(fifa19_new)
g=g.map_upper(plt.scatter)
g=g.map_lower(sns.kdeplot,camp='Blues_d')
g=g.map_diag(sns.kdeplot,lw=3,legend=False)
```

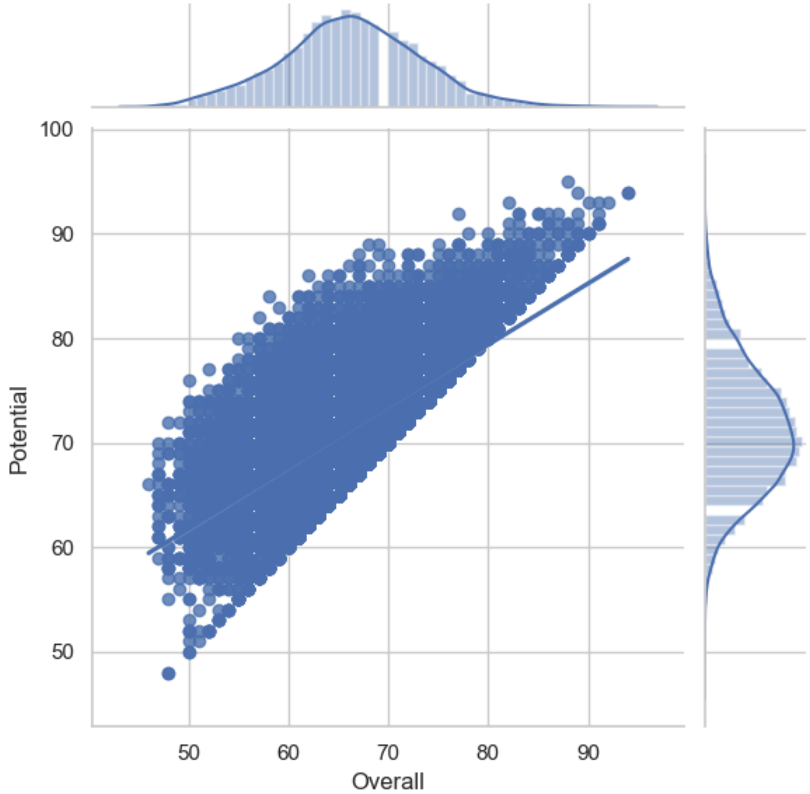


Age
Potential
Strength
Stamina
20
30
40
50
60
70
80
90
100
25
75

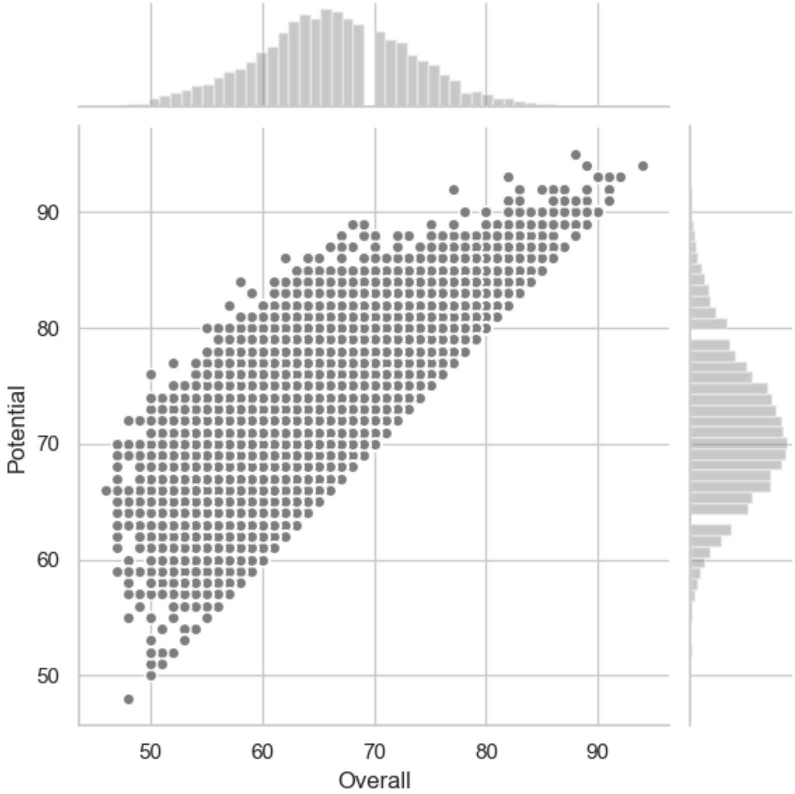
Seaborn `Jointgrid()` function

- This function provides a grid for drawing a bivariate plot with marginal univariate plots.
- It set up the grid of subplots.

In [91]:
```
# we can initialize the grid figure and add plots using default parameters as follows:-

g=sns.JointGrid(x='Overall',y='Potential',data=fifa19)
g=g.plot(sns.regplot,sns.distplot)
```

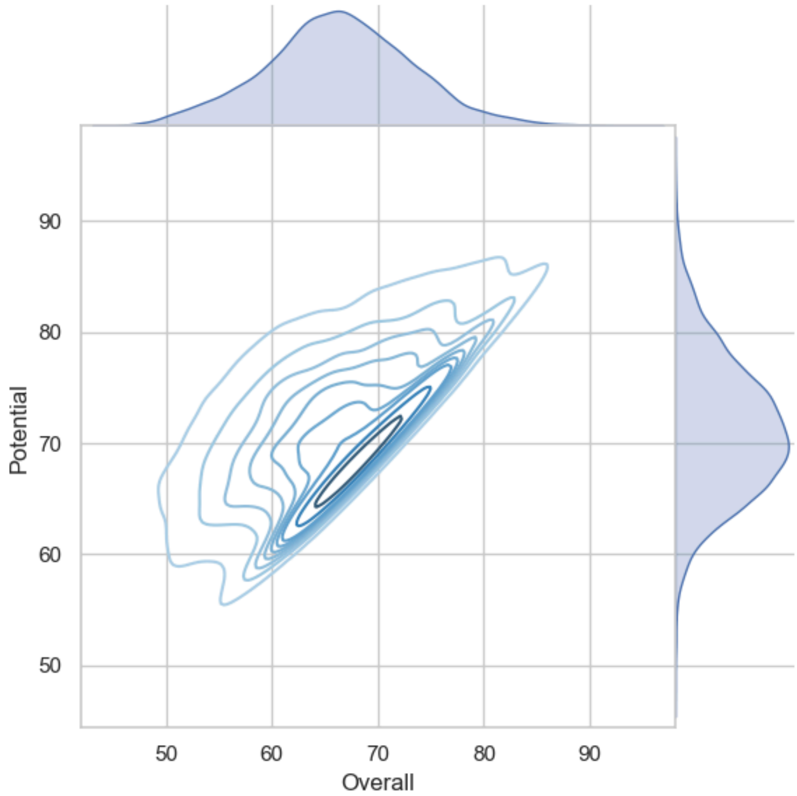


In [92]:
```
# We can draw the join and marginal plots separately, which allows finer-level control other parameters as follows:-

g = sns.JointGrid(x="Overall", y="Potential", data=fifa19)
g = g.plot_joint(plt.scatter, color=".5", edgecolor="white")
g = g.plot_marginals(sns.distplot, kde=False, color=".5")
```

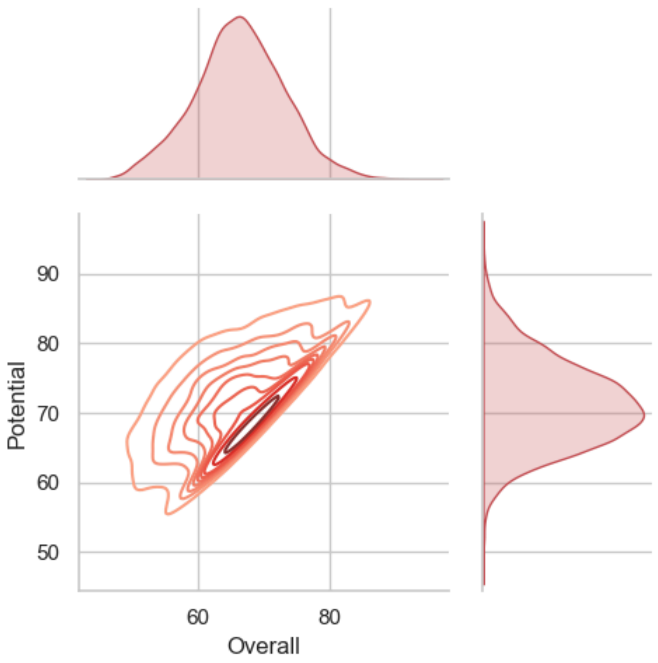


In [93]:

```
# we can remove the space between the joint and marginal axes as follows:-

g = sns.JointGrid(x="Overall", y="Potential", data=fifa19, space=0)
g = g.plot_joint(sns.kdeplot, cmap="Blues_d")
g = g.plot_marginals(sns.kdeplot, shade=True)
```



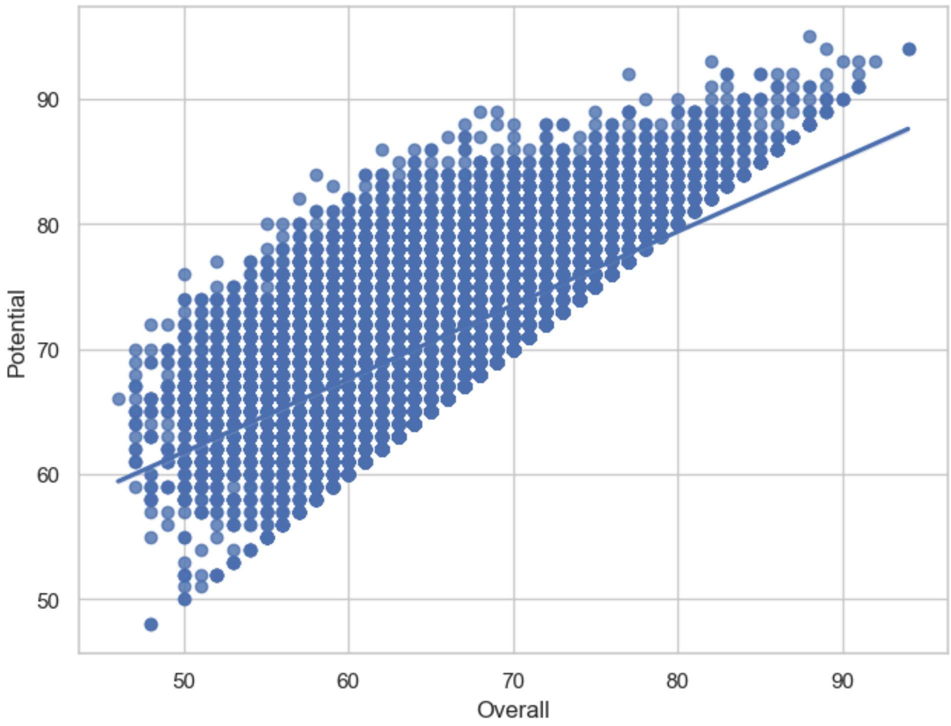
In [94]:
```
# we can draw a smaller plot with relavtively larger marginal axes as follows:-

g = sns.JointGrid(x="Overall", y="Potential", data=fifa19, 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

- The default plots made by `regplot()` and `lmplot()` look the same but on axes that have a different size and shape.
- This is because `regplot()` is an “axes-level” function draws onto a specific axes.
- This means that you can make multi-panel figures yourself and control exactly where the regression plot goes.
- If no axes object is explicitly provided, it simply uses the “currently active” axes, which is why the default plot has the same size and shape as most other matplotlib functions.
- To control the size, we need to create a figure object ourself as follows-

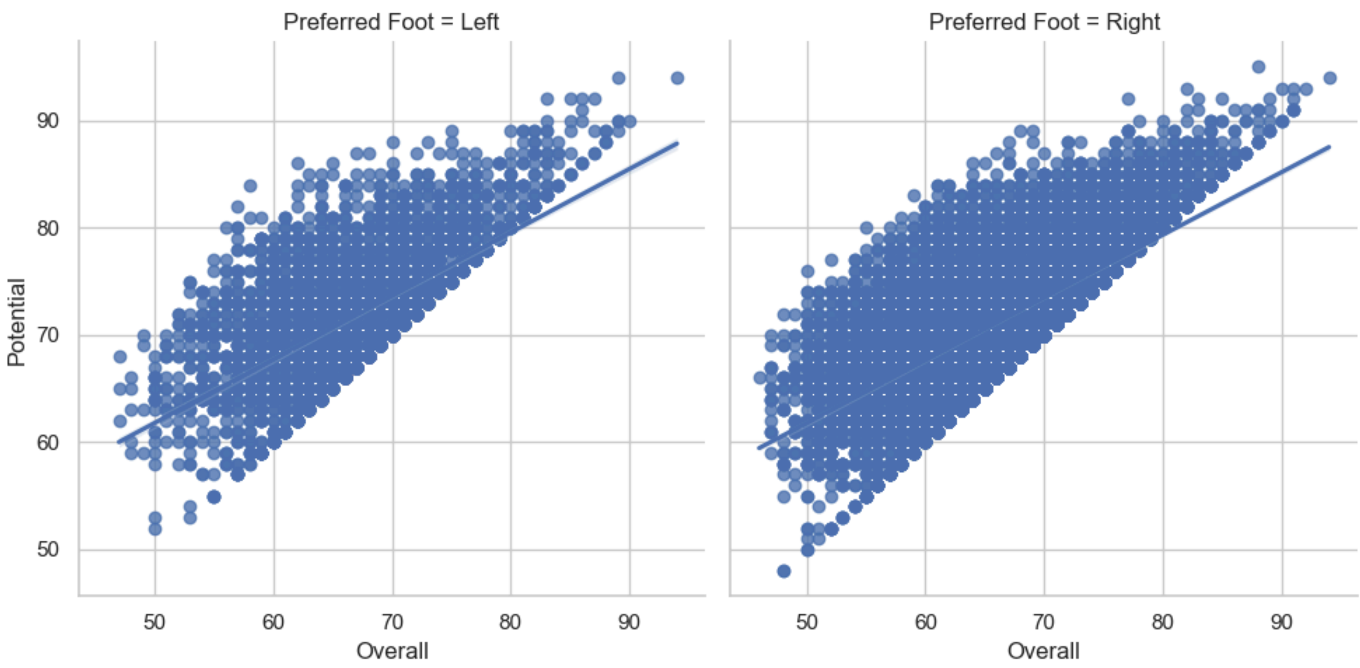
In [96]:
```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.regplot(x='Overall',y='Potential',data=fifa19);
```



In contrast, the size and shape of the `lmplot()` figure is controlled through the FacetGrid interface using the size and aspect parameters, which apply to each facet in the plot, not to the overall figure itself.

In [98]:
```
sns.lmplot(x='Overall',y='Potential',col='Preferred Foot',data=fifa19,col_wrap=2,height=5,aspect=1)
```

Out[98]:
```
<seaborn.axisgrid.FacetGrid at 0x228e1b45100>
```



Seaborn figure styles

- There are five preset seaborn themes: `darkgrid`, `whitegrid`, `dark`, `white` and `ticks`.
- They are each suited to different applications and personal preferences.
- The default theme is darkgrid.
- The grid helps the plot serve as a lookup table for quantitative information, and the white-on grey helps to keep the grid from competing with lines that represent data.
- The whitegrid theme is similar, but it is better suited to plots with heavy data elements:

I will define a simple function to plot some offset sine waves, which will help us see the different stylistic parameters as follows -

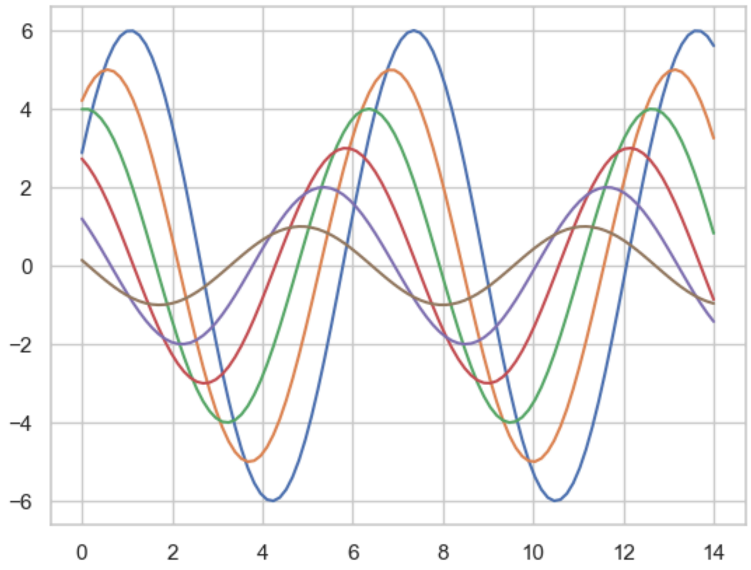
In [101…

```
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)
```

The is what the plot looks like with matplotlib default parameters

In [103…

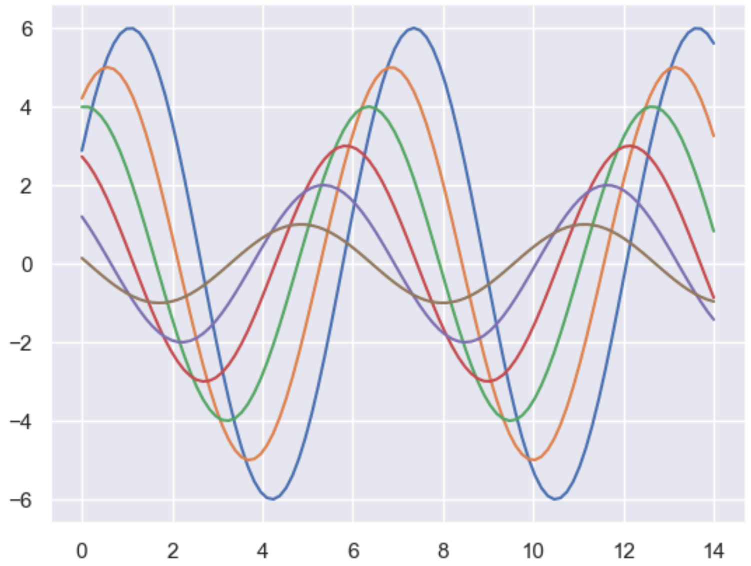
```
sinplot()
```



To switch to seaborn defaults, we need to call the 'set()' function as follows:-

In [105…

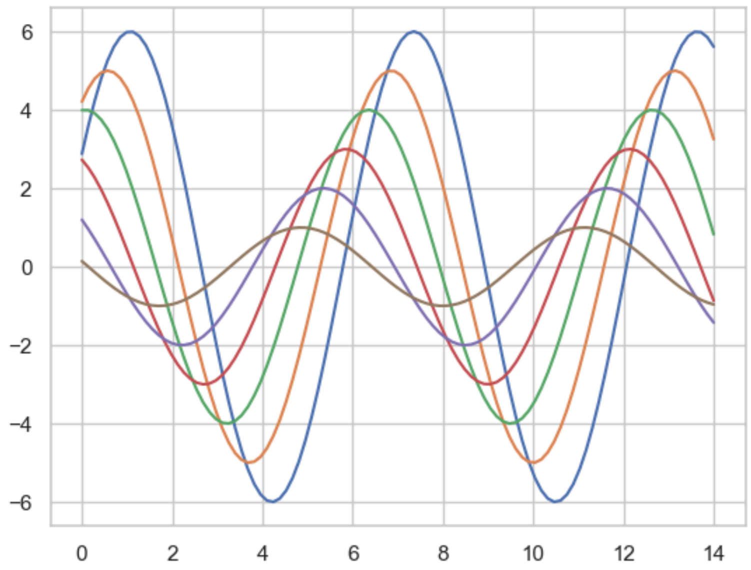
```
sns.set()
sinplot()
```



we can set different styles as follows:-

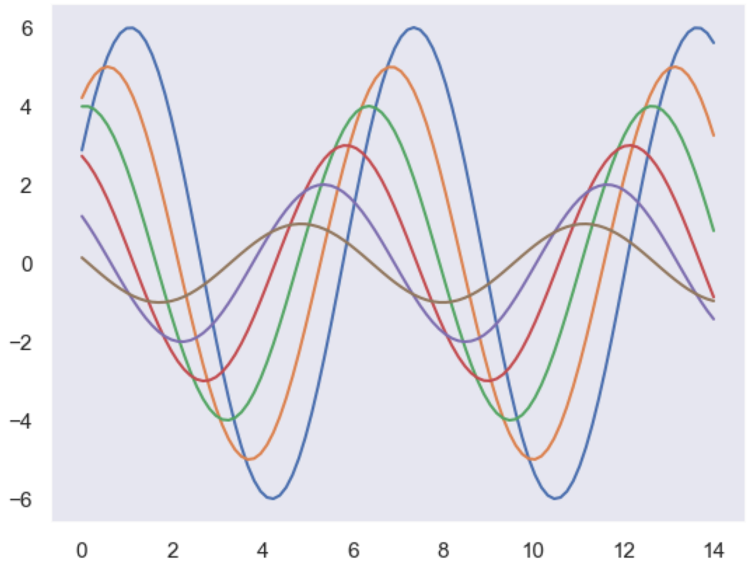
In [107…

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



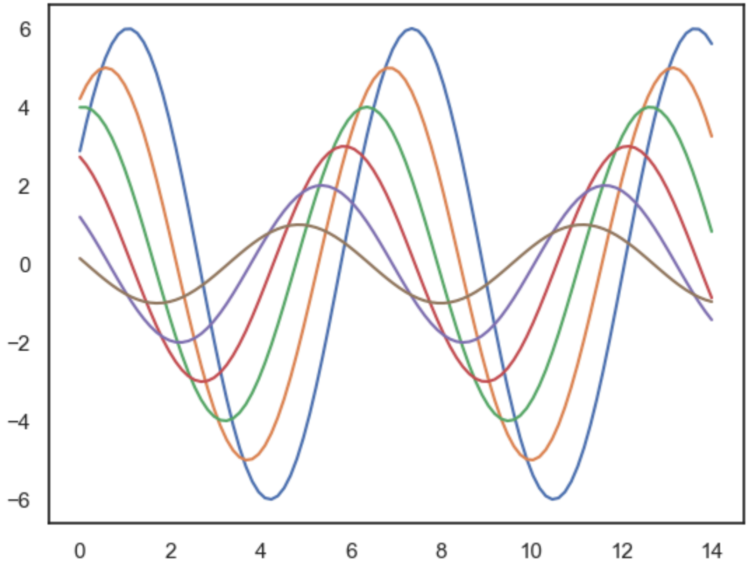
In [108…

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



In [109…

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



In [110…

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