guglielmo-basile-ikz8R6LGbK8-unsplash_1619513153581_1619513171760.jpg

In [ ]:

```
#predicting the overall rating of the players in the fifa game utilizing the  data of fifa21
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv('/content/FIFA21_official_data.csv')
df
```

Out [62]:

	ID	Name	Age	Photo	Nationality	Flag	Overall
0	176580	L. Suárez	33	https://cdn.sofifa.com/players/176/580/20_60.png	Uruguay	https://cdn.sofifa.com/flags/uy.png	87
1	192985	K. De Bruyne	29	https://cdn.sofifa.com/players/192/985/20_60.png	Belgium	https://cdn.sofifa.com/flags/be.png	91
2	212198	Bruno Fernandes	25	https://cdn.sofifa.com/players/212/198/20_60.png	Portugal	https://cdn.sofifa.com/flags/pt.png	87
3	194765	A. Griezmann	29	https://cdn.sofifa.com/players/194/765/20_60.png	France	https://cdn.sofifa.com/flags/fr.png	87
4	224334	M. Acuña	28	https://cdn.sofifa.com/players/224/334/20_60.png	Argentina	https://cdn.sofifa.com/flags/ar.png	83
...	...	...	...	...	...	...	...
17103	247866	19 C. Miszta	16	https://cdn.sofifa.com/players/247/866/19_60.png	Poland	https://cdn.sofifa.com/flags/pl.png	50
17104	251433	B. Voll	19	https://cdn.sofifa.com/players/251/433/20_60.png	Germany	https://cdn.sofifa.com/flags/de.png	51
17105	252420	T. Parker	18	https://cdn.sofifa.com/players/252/420/20_60.png	Northern Ireland	https://cdn.sofifa.com/flags/gb-nir.png	51
17106	248182	H. Sveijer	18	https://cdn.sofifa.com/players/248/182/20_60.png	Sweden	https://cdn.sofifa.com/flags/se.png	49
17107	245862	19 J. Milli	18	https://cdn.sofifa.com/players/245/862/19_60.png	Italy	https://cdn.sofifa.com/flags/it.png	47

17108 rows × 65 columns

In [ ]:

```
df.isna().sum()
```

Out [63]:

```
ID                        0
Name                      0
Age                       0
Photo                     0
Nationality               0
                       ... 
GKReflexes                0
Best Position             0
Best Overall Rating       0
Release Clause         1629
DefensiveAwareness      942
Length: 65, dtype: int64
```

In [ ]:

```
df['Club'].isna().sum()
```

Out [64]:

```
325
```

In [ ]:

```
df.head()
```

Out [65]:

	ID	Name	Age	Photo	Nationality	Flag	Overall	Po
0	176580	L. Suárez	33	https://cdn.sofifa.com/players/176/580/20_60.png	Uruguay	https://cdn.sofifa.com/flags/uy.png	87	87
1	192985	K. De Bruyne	29	https://cdn.sofifa.com/players/192/985/20_60.png	Belgium	https://cdn.sofifa.com/flags/be.png	91	91
2	212198	Bruno Fernandes	25	https://cdn.sofifa.com/players/212/198/20_60.png	Portugal	https://cdn.sofifa.com/flags/pt.png	87	90
3	194765	A. Griezmann	29	https://cdn.sofifa.com/players/194/765/20_60.png	France	https://cdn.sofifa.com/flags/fr.png	87	87
4	224334	M. Acuña	28	https://cdn.sofifa.com/players/224/334/20_60.png	Argentina	https://cdn.sofifa.com/flags/ar.png	83	83

5 rows × 65 columns

In [ ]:

```
df.tail()
```

Out [66]:

	ID	Name	Age	Photo	Nationality	Flag	Overall	P
17103	247866	19 C. Miszta	16	https://cdn.sofifa.com/players/247/866/19_60.png	Poland	https://cdn.sofifa.com/flags/pl.png	50	7
17104	251433	B. Voll	19	https://cdn.sofifa.com/players/251/433/20_60.png	Germany	https://cdn.sofifa.com/flags/de.png	51	6
17105	252420	T. Parker	18	https://cdn.sofifa.com/players/252/420/20_60.png	Northern Ireland	https://cdn.sofifa.com/flags/gb-nir.png	51	7
17106	248182	H. Sveijer	18	https://cdn.sofifa.com/players/248/182/20_60.png	Sweden	https://cdn.sofifa.com/flags/se.png	49	6
17107	245862	19 J. Milli	18	https://cdn.sofifa.com/players/245/862/19_60.png	Italy	https://cdn.sofifa.com/flags/it.png	47	6

5 rows × 65 columns

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

Out [67]:

	ID	Age	Overall	Potential	Special	International Reputation	Weak Foot	Skill Moves	
count	17108.000000	17108.000000	17108.000000	17108.000000	17108.000000	17108.000000	17108.000000	17108.000000	1708
mean	221421.276187	25.053718	66.780161	72.553542	1625.722995	1.147533	2.981938	2.446107	20.75
std	36028.786065	4.915963	7.019069	5.738347	263.503922	0.455773	0.674699	0.780278	17.19
min	2.000000	16.000000	38.000000	46.000000	731.000000	1.000000	1.000000	1.000000	1.000
25%	205451.750000	21.000000	62.000000	69.000000	1484.000000	1.000000	3.000000	2.000000	9.000
50%	230441.000000	24.000000	67.000000	72.000000	1653.000000	1.000000	3.000000	2.000000	18.00
75%	245402.500000	28.000000	72.000000	76.000000	1810.000000	1.000000	3.000000	3.000000	27.00
max	259105.000000	53.000000	93.000000	95.000000	2316.000000	5.000000	5.000000	5.000000	99.00

8 rows × 45 columns

In [ ]:
```
df.dtypes
```

Out [68]:
```
ID                          int64
Name                       object
Age                         int64
Photo                      object
Nationality                object
                           ...
GKReflexes                float64
Best Position              object
Best Overall Rating       float64
Release Clause             object
DefensiveAwareness        float64
Length: 65, dtype: object
```

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

Out [69]:
```
(17108, 65)
```

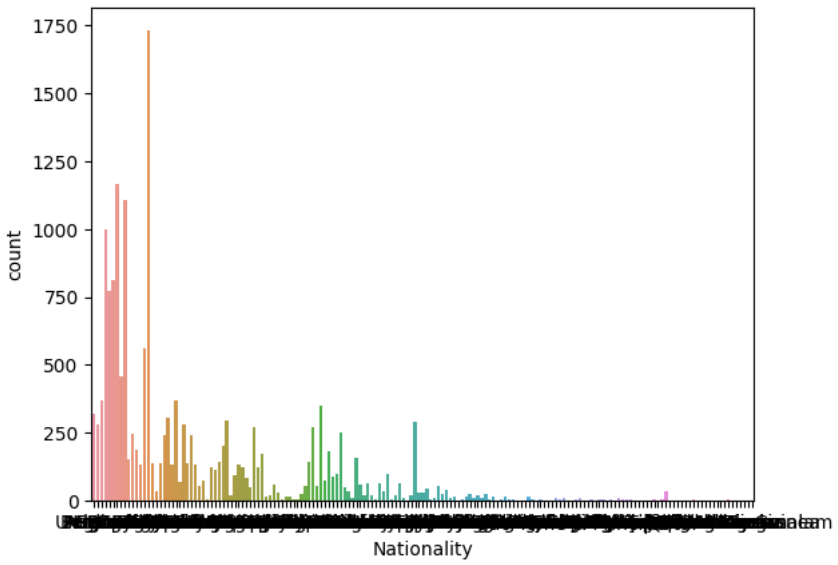
In [ ]:
```
newdf=df['Nationality'].value_counts()
newdf
```

Out [70]:
```
England        1730
Germany        1166
Spain          1106
France          997
Brazil          811
               ...
Tanzania          1
Rwanda            1
Suriname          1
Puerto Rico       1
Oman              1
Name: Nationality, Length: 169, dtype: int64
```

In [ ]:
```
# This visualization helps us gain insights into the composition of players from different countries in the

sns.countplot(x='Nationality',data=df)
```

Out [71]:
```
<Axes: xlabel='Nationality', ylabel='count'>
```



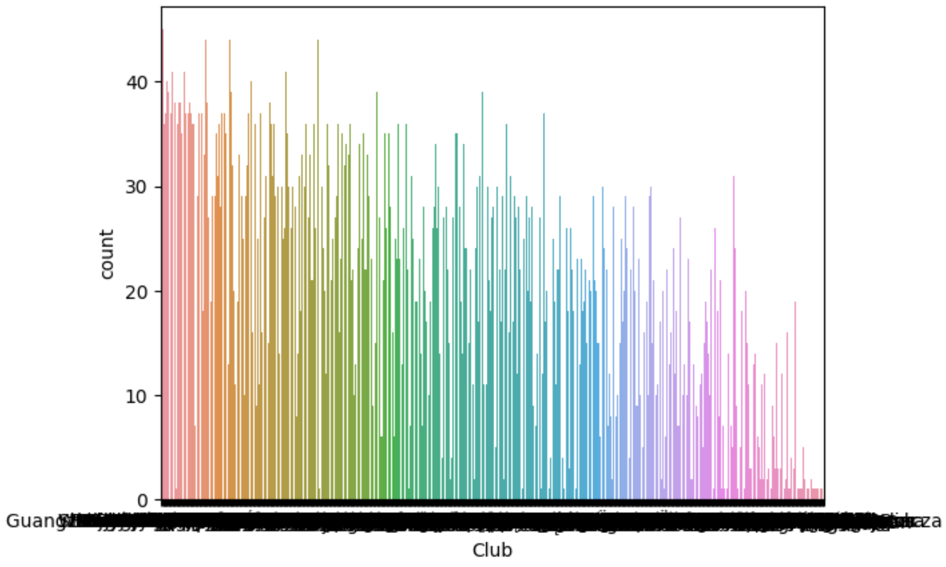
In [ ]:
```
nd=df['Club'].value_counts()
nd
```

Out [72]:
```
Manchester United                  45
Bolton Wanderers                   44
Crystal Palace                     44
Arsenal                            44
AS Monaco                          44
                                   ..
Associação Académica de Coimbra     1
Carpi                               1
SC Fortuna Köln                     1
Siena                               1
Sakaryaspor                         1
Name: Club, Length: 843, dtype: int64
```

In [ ]:
```
# countplot of player distribution among clubs

sns.countplot(x='Club',data=df)
```

Out [73]:
```
<Axes: xlabel='Club', ylabel='count'>
```



In [ ]:
```
nd=df['Potential'].value_counts()
nd
```

Out [74]:
```
72    1321
73    1277
71    1177
75    1167
70    1161
74    1143
69     953
76     863
68     859
67     761
77     754
78     696
66     620
79     560
65     539
80     484
```

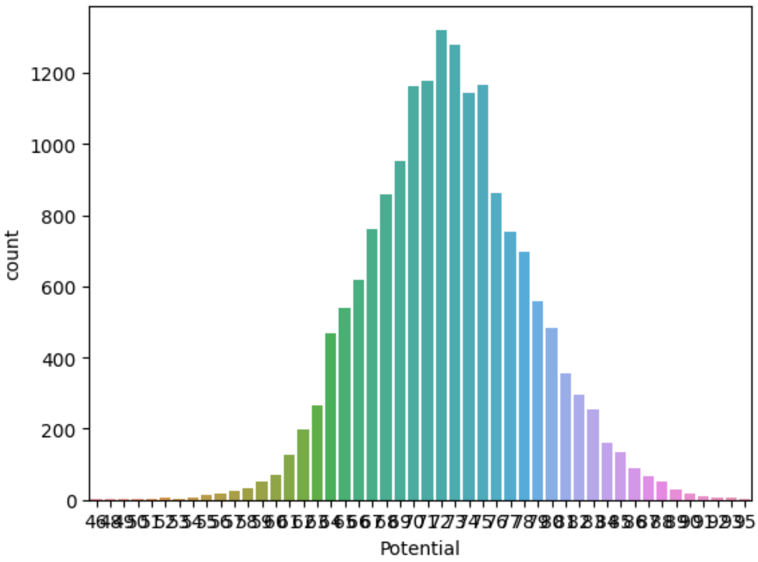
```
64      469
81      357
82      295
63      264
83      256
62      196
84      161
85      134
61      127
86       90
60       72
87       68
88       52
59       52
58       34
89       29
57       24
90       18
56       16
55       14
91       12
93        7
92        5
52        5
54        5
50        3
48        3
95        1
49        1
46        1
51        1
53        1
Name: Potential, dtype: int64
```

In [ ]:

```
# identify the prevalence of different potential levels among players in the game


sns.countplot(x='Potential',data=df)
```

Out [75]: `<Axes: xlabel='Potential', ylabel='count'>`



In [ ]:

```
nd=df['Best Position'].value_counts()
nd
```

Out [76]:

```
CB      3219
ST      2652
CAM     2255
GK      1603
RM      1432
CDM     1260
CM       963
RB       914
LB       907
LM       814
RW       308
RWB      251
LWB      250
LW       196
CF        84
Name: Best Position, dtype: int64
```

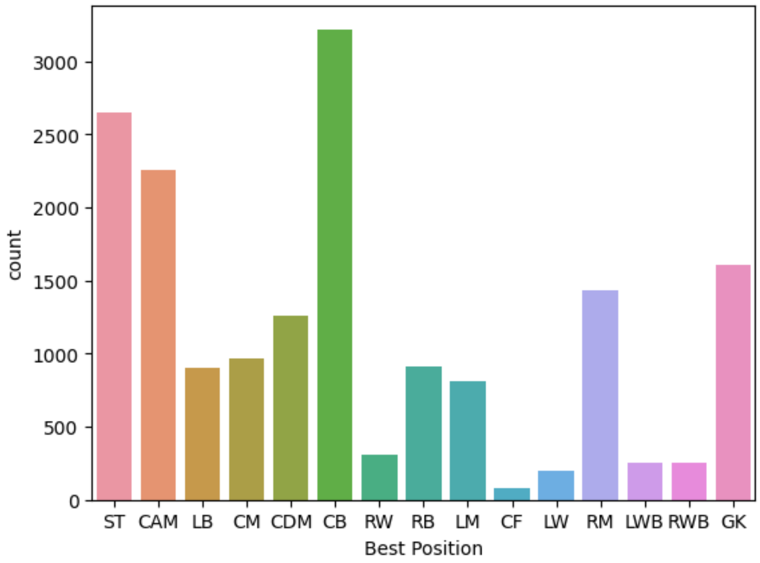
In [ ]:

```
# This visualization serves as a fundamental resource for understanding the distribution of player positions



sns.countplot(x='Best Position',data=df)
```

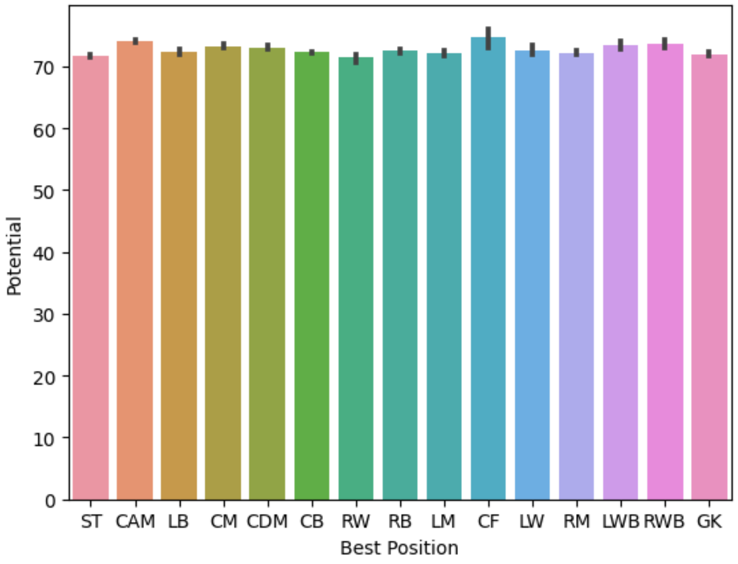
```
Out [77]: <Axes: xlabel='Best Position', ylabel='count'>
```



```
In [ ]: # A barplot is an ideal choice for this analysis because it allows us to compare the average potential ratir

        sns.barplot(x='Best Position',y='Potential',data=df)
```

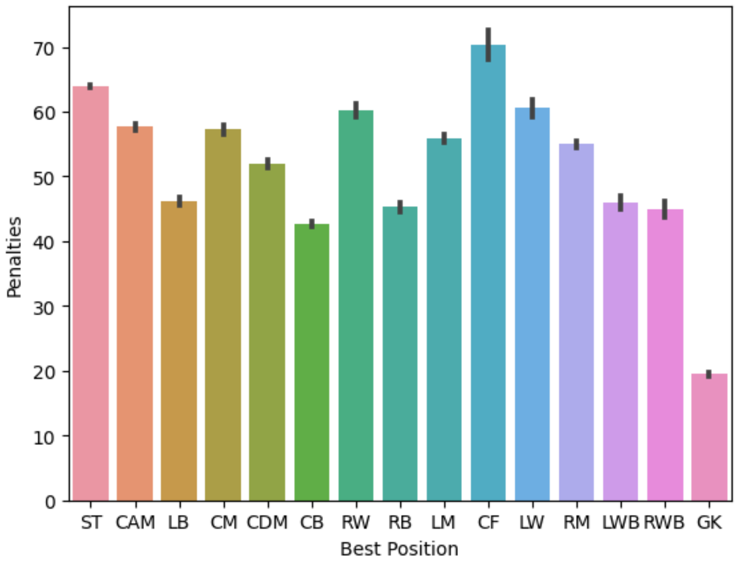
```
Out [78]: <Axes: xlabel='Best Position', ylabel='Potential'>
```



```
In [ ]: 
        # compare the average penalty-taking ratings (y-axis) for each unique best playing position (x-axis)

        sns.barplot(x='Best Position',y='Penalties',data=df)
```

```
Out [79]: <Axes: xlabel='Best Position', ylabel='Penalties'>
```



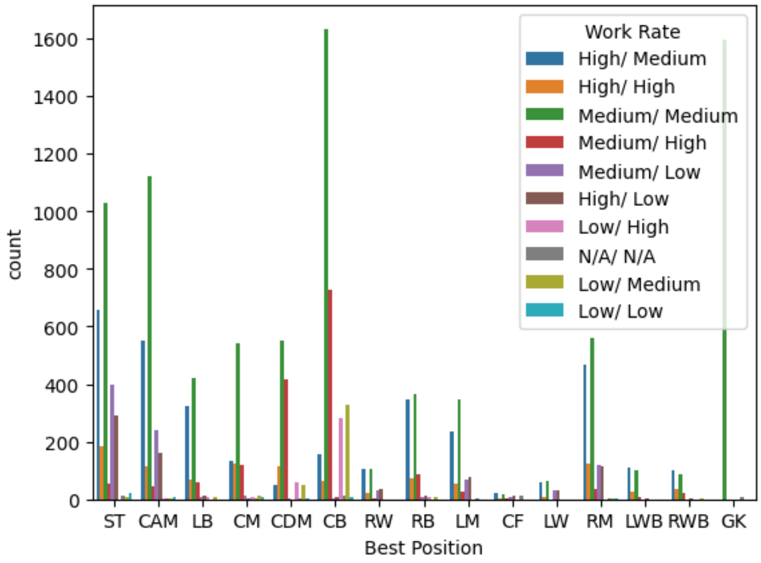
In [ ]:

```
# it counts and represents the occurrences of each unique best playing position (x-axis) while further disti

sns.countplot(x='Best Position',data=df,hue='Work Rate')
```

Out [80]:

```
<Axes: xlabel='Best Position', ylabel='count'>
```



In [ ]:

```
df1=pd.get_dummies(df[['Nationality','Work Rate','Club','Best Position','Preferred Foot']],drop_first=True)
df1
```

Out [81]:

	Nationality_Albania	Nationality_Algeria	Nationality_Andorra	Nationality_Angola	Nationality_Antigua & Barbuda	Nationality_Argentina	Nationa
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	1	0
...	...	...	...	...	...	...	...
17103	0	0	0	0	0	0	0
17104	0	0	0	0	0	0	0
17105	0	0	0	0	0	0	0
17106	0	0	0	0	0	0	0
17107	0	0	0	0	0	0	0

17108 rows × 1034 columns

In [ ]:

```
dfe=pd.concat([df,df1],axis=1)
dfe
```

Out [82]:

	ID	Name	Age	Photo	Nationality	Flag	Overall
0	176580	L. Suárez	33	https://cdn.sofifa.com/players/176/580/20_60.png	Uruguay	https://cdn.sofifa.com/flags/uy.png	87
1	192985	K. De Bruyne	29	https://cdn.sofifa.com/players/192/985/20_60.png	Belgium	https://cdn.sofifa.com/flags/be.png	91
2	212198	Bruno Fernandes	25	https://cdn.sofifa.com/players/212/198/20_60.png	Portugal	https://cdn.sofifa.com/flags/pt.png	87
3	194765	A. Griezmann	29	https://cdn.sofifa.com/players/194/765/20_60.png	France	https://cdn.sofifa.com/flags/fr.png	87
4	224334	M. Acuña	28	https://cdn.sofifa.com/players/224/334/20_60.png	Argentina	https://cdn.sofifa.com/flags/ar.png	83
...	...	...	...	...	...	...	...
17103	247866	19 C. Miszta	16	https://cdn.sofifa.com/players/247/866/19_60.png	Poland	https://cdn.sofifa.com/flags/pl.png	50
17104	251433	B. Voll	19	https://cdn.sofifa.com/players/251/433/20_60.png	Germany	https://cdn.sofifa.com/flags/de.png	51
17105	252420	T. Parker	18	https://cdn.sofifa.com/players/252/420/20_60.png	Northern Ireland	https://cdn.sofifa.com/flags/gb-nir.png	51
17106	248182	H. Sveijer	18	https://cdn.sofifa.com/players/248/182/20_60.png	Sweden	https://cdn.sofifa.com/flags/se.png	49
17107	245862	19 J. Milli	18	https://cdn.sofifa.com/players/245/862/19_60.png	Italy	https://cdn.sofifa.com/flags/it.png	47

17108 rows × 1099 columns

In [ ]:

```
dfe=dfe.drop(['Club','Name','Nationality','Work Rate','Best Position','ID','Photo','Flag','Club Logo','Speci
dfe
```

Out [83]:

	Age	Overall	Potential	Value	Wage	Weak Foot	Skill Moves	Height	Weight	Crossing	...	Best Position_LB	Best Position_LM	Best Position_LW
0	33	87	87	€31.5M	€115K	4.0	3.0	6'0	190lbs	80.0	...	0	0	0
1	29	91	91	€87M	€370K	5.0	4.0	5'11	154lbs	94.0	...	0	0	0
2	25	87	90	€63M	€195K	4.0	4.0	5'10	152lbs	87.0	...	0	0	0
3	29	87	87	€50.5M	€290K	3.0	4.0	5'9	161lbs	83.0	...	0	0	0
4	28	83	83	€22M	€41K	3.0	4.0	5'8	152lbs	87.0	...	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
17103	16	50	70	€50K	€500	3.0	1.0	6'4	176lbs	14.0	...	0	0	0
17104	19	51	63	€50K	€500	2.0	1.0	6'5	187lbs	8.0	...	0	0	0
17105	18	51	70	€60K	€500	3.0	1.0	6'3	176lbs	10.0	...	0	0	0
17106	18	49	63	€50K	€500	2.0	1.0	6'1	168lbs	10.0	...	0	0	0
17107	18	47	65	€50K	€500	3.0	1.0	6'0	172lbs	10.0	...	0	0	0

17108 rows × 1078 columns

In [ ]:

```
dfe['Value']=dfe['Value'].str.replace('€', '')
dfe['Value']=dfe['Value'].str.replace('M', '')
dfe['Value']=dfe['Value'].str.replace('K', '')
dfe
```

Out [84]:

	Age	Overall	Potential	Value	Wage	Weak Foot	Skill Moves	Height	Weight	Crossing	...	Best Position_LB	Best Position_LM	Best Position_LW	Po
0	33	87	87	31.5	€115K	4.0	3.0	6'0	190lbs	80.0	...	0	0	0	0
1	29	91	91	87	€370K	5.0	4.0	5'11	154lbs	94.0	...	0	0	0	0
2	25	87	90	63	€195K	4.0	4.0	5'10	152lbs	87.0	...	0	0	0	0
3	29	87	87	50.5	€290K	3.0	4.0	5'9	161lbs	83.0	...	0	0	0	0
4	28	83	83	22	€41K	3.0	4.0	5'8	152lbs	87.0	...	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
17103	16	50	70	50	€500	3.0	1.0	6'4	176lbs	14.0	...	0	0	0	0
17104	19	51	63	50	€500	2.0	1.0	6'5	187lbs	8.0	...	0	0	0	0
17105	18	51	70	60	€500	3.0	1.0	6'3	176lbs	10.0	...	0	0	0	0
17106	18	49	63	50	€500	2.0	1.0	6'1	168lbs	10.0	...	0	0	0	0
17107	18	47	65	50	€500	3.0	1.0	6'0	172lbs	10.0	...	0	0	0	0

17108 rows × 1078 columns

In [ ]:

```
dfe['Wage']=dfe['Wage'].str.replace('€', '')
dfe['Wage']=dfe['Wage'].str.replace('K', '')
```

```
dfe
```

Out [85]:

	Age	Overall	Potential	Value	Wage	Weak Foot	Skill Moves	Height	Weight	Crossing	...	Best Position_LB	Best Position_LM	Best Position_LW	Pos
0	33	87	87	31.5	115	4.0	3.0	6'0	190lbs	80.0	...	0	0	0	0
1	29	91	91	87	370	5.0	4.0	5'11	154lbs	94.0	...	0	0	0	0
2	25	87	90	63	195	4.0	4.0	5'10	152lbs	87.0	...	0	0	0	0
3	29	87	87	50.5	290	3.0	4.0	5'9	161lbs	83.0	...	0	0	0	0
4	28	83	83	22	41	3.0	4.0	5'8	152lbs	87.0	...	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
17103	16	50	70	50	500	3.0	1.0	6'4	176lbs	14.0	...	0	0	0	0
17104	19	51	63	50	500	2.0	1.0	6'5	187lbs	8.0	...	0	0	0	0
17105	18	51	70	60	500	3.0	1.0	6'3	176lbs	10.0	...	0	0	0	0
17106	18	49	63	50	500	2.0	1.0	6'1	168lbs	10.0	...	0	0	0	0
17107	18	47	65	50	500	3.0	1.0	6'0	172lbs	10.0	...	0	0	0	0

17108 rows × 1078 columns

In [ ]:

```
dfe['Height']=dfe['Height'].str.replace("'",".")
dfe
```

Out [86]:

	Age	Overall	Potential	Value	Wage	Weak Foot	Skill Moves	Height	Weight	Crossing	...	Best Position_LB	Best Position_LM	Best Position_LW	Pos
0	33	87	87	31.5	115	4.0	3.0	6.0	190lbs	80.0	...	0	0	0	0
1	29	91	91	87	370	5.0	4.0	5.11	154lbs	94.0	...	0	0	0	0
2	25	87	90	63	195	4.0	4.0	5.10	152lbs	87.0	...	0	0	0	0
3	29	87	87	50.5	290	3.0	4.0	5.9	161lbs	83.0	...	0	0	0	0
4	28	83	83	22	41	3.0	4.0	5.8	152lbs	87.0	...	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
17103	16	50	70	50	500	3.0	1.0	6.4	176lbs	14.0	...	0	0	0	0
17104	19	51	63	50	500	2.0	1.0	6.5	187lbs	8.0	...	0	0	0	0
17105	18	51	70	60	500	3.0	1.0	6.3	176lbs	10.0	...	0	0	0	0
17106	18	49	63	50	500	2.0	1.0	6.1	168lbs	10.0	...	0	0	0	0
17107	18	47	65	50	500	3.0	1.0	6.0	172lbs	10.0	...	0	0	0	0

17108 rows × 1078 columns

In [ ]:

```
dfe['Weight']=dfe['Weight'].str.replace("lbs"," ")
dfe
```

Out [87]:

	Age	Overall	Potential	Value	Wage	Weak Foot	Skill Moves	Height	Weight	Crossing	...	Best Position_LB	Best Position_LM	Best Position_LW	Pos
0	33	87	87	31.5	115	4.0	3.0	6.0	190	80.0	...	0	0	0	0
1	29	91	91	87	370	5.0	4.0	5.11	154	94.0	...	0	0	0	0
2	25	87	90	63	195	4.0	4.0	5.10	152	87.0	...	0	0	0	0
3	29	87	87	50.5	290	3.0	4.0	5.9	161	83.0	...	0	0	0	0
4	28	83	83	22	41	3.0	4.0	5.8	152	87.0	...	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
17103	16	50	70	50	500	3.0	1.0	6.4	176	14.0	...	0	0	0	0
17104	19	51	63	50	500	2.0	1.0	6.5	187	8.0	...	0	0	0	0
17105	18	51	70	60	500	3.0	1.0	6.3	176	10.0	...	0	0	0	0
17106	18	49	63	50	500	2.0	1.0	6.1	168	10.0	...	0	0	0	0
17107	18	47	65	50	500	3.0	1.0	6.0	172	10.0	...	0	0	0	0

17108 rows × 1078 columns

In [ ]:

```
dfe.dtypes
```

Out [88]:

```
Age                       int64
Overall                   int64
Potential                 int64
Value                    object
Wage                     object
                          ...
Best Position_RM          uint8
Best Position_RW          uint8
Best Position_RWB         uint8
Best Position_ST         uint8
Preferred Foot_Right     uint8
Length: 1078, dtype: object
```

In [ ]:

```
dfe['Wage']=dfe['Wage'].astype(float)
dfe['Value']=dfe['Value'].astype(float)
dfe['Height']=dfe['Height'].astype(float)
dfe['Weight']=dfe['Weight'].astype(float)
dfe.dtypes
```

Out [89]:

```
Age                       int64
Overall                   int64
Potential                 int64
Value                   float64
Wage                    float64
                         ...   
Best Position_RM          uint8
Best Position_RW          uint8
Best Position_RWB         uint8
Best Position_ST          uint8
Preferred Foot_Right      uint8
Length: 1078, dtype: object
```

In [ ]:

```
dfe.isna().sum()
```

Out [90]:

```
Age                     0
Overall                 0
Potential               0
Value                   0
Wage                    0
                       ..
Best Position_RM        0
Best Position_RW        0
Best Position_RWB       0
Best Position_ST        0
Preferred Foot_Right    0
Length: 1078, dtype: int64
```

In [ ]:

```
dfe['DefensiveAwareness'].isna().sum()
```

Out [91]:

```
942
```

In [ ]:

```
dfe.loc[dfe.Value==0,'Value']=np.NAN
dfe.loc[dfe.Wage==0,'Wage']=np.NAN
dfe.isna().sum()
```

Out [92]:

```
Age                       0
Overall                   0
Potential                 0
Value                   409
Wage                    380
                       ... 
Best Position_RM          0
Best Position_RW          0
Best Position_RWB         0
Best Position_ST          0
Preferred Foot_Right      0
Length: 1078, dtype: int64
```

In [ ]:

```
dfe['Value']=dfe['Value'].fillna(dfe['Value'].mean())
dfe['DefensiveAwareness']=dfe['DefensiveAwareness'].fillna(dfe['DefensiveAwareness'].mode()[0])
dfe['Wage']=dfe['Wage'].fillna(dfe['Wage'].mean())
dfe.isna().sum()
```

Out [93]:

```
Age                     0
Overall                 0
Potential               0
Value                   0
Wage                    0
                       ..
Best Position_RM        0
Best Position_RW        0
Best Position_RWB       0
Best Position_ST        0
Preferred Foot_Right    0
Length: 1078, dtype: int64
```

In [ ]:

```
dfe= dfe.fillna(0)
```

In [ ]:

```
x=dfe.drop(['Overall'],axis=1)
y=dfe['Overall']
```

In [ ]:

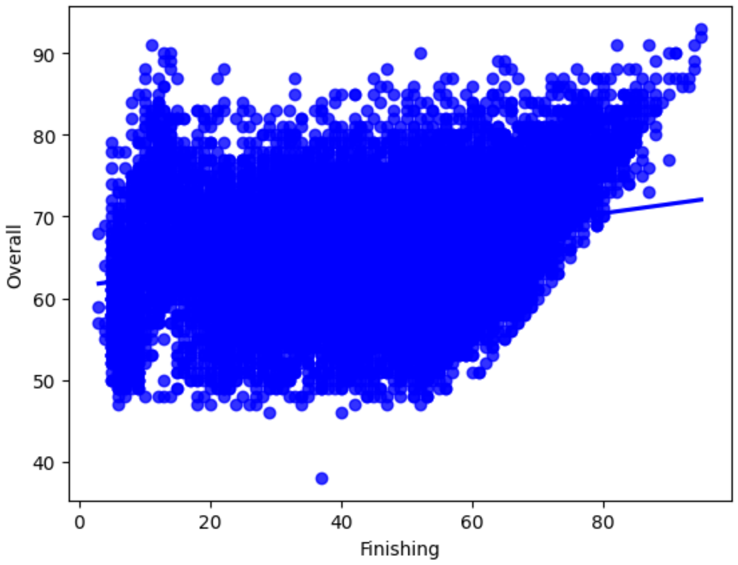
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=.30,random_state=42)
```

In [ ]:

```
sns.regplot(x=dfe['Finishing'],y=y,color='blue')
```

Out [97]:

```
<Axes: xlabel='Finishing', ylabel='Overall'>
```

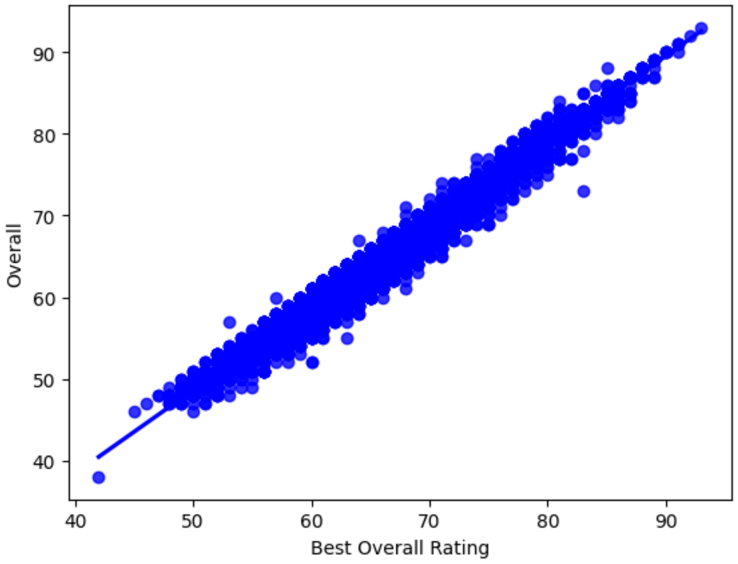


In [ ]:

```
sns.regplot(x=dfe['Best Overall Rating'],y=y,color='blue')
```

Out [98]:

```
<Axes: xlabel='Best Overall Rating', ylabel='Overall'>
```



SIMPLE LINEAR REGRESSION

In [ ]:

```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(xtrain,ytrain)
ypred=model.predict(xtest)
ypred
```

Out [99]:

```
array([70.52409475, 65.18623825, 71.30323951, ..., 71.89032842,
       70.3503439 , 68.60023982])
```

In [ ]:

```
from sklearn.metrics import mean_absolute_error
print('mae',mean_absolute_error(ytest,ypred))
```

```
mae 0.6504289769623379
```

In [ ]:

```
from sklearn.metrics import mean_absolute_percentage_error
print("error percentage is ",mean_absolute_percentage_error(ytest,ypred))
```

```
error percentage is  0.009903543857174927
```

In [ ]:

```
from sklearn.metrics import mean_squared_error
print("mse is ",mean_squared_error(ytest,ypred))
```

```
mse is  0.7838686957187891
```

In [ ]:

```
numbu=mean_squared_error(ytest,ypred)
np.sqrt(numbu)
```

```
Out [103]: 0.8853635952075222
```

```
In [ ]: from sklearn.metrics import r2_score
        slr2=r2_score(ytest,ypred)
        print('r2 score is',slr2)
```

```
r2 score is 0.984032392127196
```

Random Forest

```
In [ ]: from sklearn.ensemble import RandomForestRegressor
        reg = RandomForestRegressor()
        reg.fit(xtrain, ytrain)
```

```
Out [105]: ▾ RandomForestRegressor
           RandomForestRegressor()
```

```
In [ ]: reg.score(xtest, ytest)
```

```
Out [106]: 0.9950703724690045
```

```
In [ ]: reg.score(xtrain, ytrain)
```

```
Out [107]: 0.9992168538270579
```

```
In [ ]: ypred2= reg.predict(xtest)
        ypred2
```

```
Out [108]: array([71.  , 64.92, 71.8 , ..., 72.  , 71.91, 67.95])
```

```
In [ ]: from sklearn.metrics import mean_absolute_error
        print('mae',mean_absolute_error(ytest,ypred2))
```

```
mae 0.27849795441262426
```

```
In [ ]: from sklearn.metrics import mean_absolute_percentage_error
        print("error percentage is ",mean_absolute_percentage_error(ytest,ypred2))
```

```
error percentage is  0.004309717958739423
```

```
In [ ]: from sklearn.metrics import mean_squared_error
        print("mse is ",mean_squared_error(ytest,ypred2))
```

```
mse is  0.24200122735242552
```

```
In [ ]: numbu=mean_squared_error(ytest,ypred2)
        np.sqrt(numbu)
```

```
Out [112]: 0.4919362025226701
```

```
In [ ]: from sklearn.metrics import r2_score
        rfr2=r2_score(ytest,ypred2)
        print('r2 score is',rfr2)
```

```
r2 score is 0.9950703724690045
```

Decision tree

```
In [ ]: from sklearn.tree import DecisionTreeRegressor
        dtr= DecisionTreeRegressor()
        dtr.fit(xtrain,ytrain)
```

```
Out [114]: ▾ DecisionTreeRegressor
           DecisionTreeRegressor()
```

```
In [ ]: dtr.score(xtest, ytest)
```

```
Out [115]: 0.9897573277040124
```

```
In [ ]: ypred3=dtr.predict(xtest)
```

```
In [ ]: ypred3
```

```
Out [117]: array([71., 65., 71., ..., 72., 72., 68.])
```

```
In [ ]: from sklearn.metrics import mean_absolute_error
        print('mae',mean_absolute_error(ytest,ypred3))
```

```
mae 0.30995519189557763
```

```
In [ ]: from sklearn.metrics import mean_absolute_percentage_error
        print("error percentage is ",mean_absolute_percentage_error(ytest,ypred3))
```

```
error percentage is  0.004825143164415268
```

```
In [ ]: from sklearn.metrics import mean_squared_error
        print("mse is ",mean_squared_error(ytest,ypred3))
```

```
mse is  0.5028248587570622
```

```
In [ ]: from sklearn.metrics import mean_squared_error
        print("mse is ",mean_squared_error(ytest,ypred3))
```

```
mse is  0.5028248587570622
```

```
In [ ]: from sklearn.metrics import r2_score
        dtr2=r2_score(ytest,ypred3)
        print('r2 score is',dtr2)
```

```
r2 score is 0.9897573277040124
```

CONCLUSION

all the above models have same accuracy.however random forest regression model is slightly better than the other three models