In [50]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from statsmodels.distributions.empirical_distribution import ECDF # Empiri
```

# Cricket

In [2]:

```
sehwag = pd.read_csv("sehwag.csv")
dravid = pd.read_csv("dravid.csv")
```

In [3]:

```
sehwag
```

Out[3]:

| | Runs | Mins | BF | 4s | 6s | SR | Pos | Dismissal | Inns | Unnamed: 9 | Opposition | Gro |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 5 | 2 | 0 | 0 | 50.00 | 7 | lbw | 1 | NaN | v Pakistan | M |
| **1** | 19 | 18 | 24 | 0 | 1 | 79.16 | 6 | caught | 1 | NaN | v Zimbabwe | R |
| **2** | 58 | 62 | 54 | 8 | 0 | 107.40 | 6 | bowled | 1 | NaN | v Australia | Beng |
| **3** | 2 | 7 | 7 | 0 | 0 | 28.57 | 6 | caught | 2 | NaN | v Zimbabwe | Bula |
| **4** | 11 | 19 | 16 | 1 | 0 | 68.75 | 6 | not out | 2 | NaN | v West Indies | Bula |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **240** | 15 | 21 | 15 | 2 | 0 | 100.00 | 2 | caught | 1 | NaN | v Sri Lanka | Hamba |
| **241** | 3 | 6 | 6 | 0 | 0 | 50.00 | 2 | caught | 2 | NaN | v Sri Lanka | Colo ( |
| **242** | 34 | 46 | 29 | 6 | 0 | 117.24 | 2 | caught | 2 | NaN | v Sri Lanka | Colo ( |
| **243** | 4 | 20 | 11 | 1 | 0 | 36.36 | 2 | bowled | 1 | NaN | v Pakistan | Che |
| **244** | 31 | 70 | 43 | 3 | 0 | 72.09 | 2 | lbw | 2 | NaN | v Pakistan | Ko |

245 rows × 14 columns

In [4]:

```
sehwag.shape
```

Out[4]:

```
(245, 14)
```

In [5]:

```
sehwag["Runs"].describe()
```

```
Out[5]: count    245.000000
        mean      33.767347
        std       34.809419
        min        0.000000
        25%        8.000000
        50%       23.000000
        75%       46.000000
        max      219.000000
        Name: Runs, dtype: float64
```

```
In [6]: p_25 = np.percentile(sehwag["Runs"], 25) # 25th percentile or Q1
        p_25
```

```
Out[6]: 8.0
```

```
In [7]: p_50 = np.percentile(sehwag["Runs"], 50) # 50th percentile or Q2, "median"
        p_50
```

```
Out[7]: 23.0
```

```
In [8]: p_75 = np.percentile(sehwag["Runs"], 75) # 75th percentile or Q3
        p_75
```

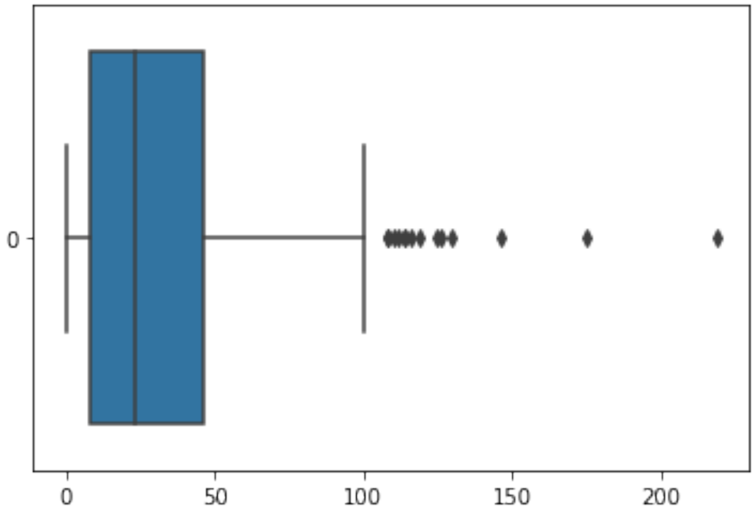
```
Out[8]: 46.0
```

```
In [9]: iqr = p_75 - p_25 # Inter quartile range
        iqr
```

```
Out[9]: 38.0
```

```
In [10]: sns.boxplot(data=sehwag["Runs"], orient="h")
```

```
Out[10]: <AxesSubplot:>
```



```
In [11]: lower = max(p_25 - 1.5 * iqr, 0)
         lower
```

```
Out[11]: 0
```

In [12]:

```
upper = p_75 + 1.5 * iqr
upper
```

Out[12]:

```
103.0
```

In [13]:

```
sehwag_outlier = sehwag[sehwag["Runs"] > upper]
```

In [14]:

```
len(sehwag_outlier)
```

Out[14]:

```
14
```

In [15]:

```
14/245
```

Out[15]:

```
0.05714285714285714
```

In [19]:

```
dravid["Runs"].describe()
```

Out[19]:

```
count    318.000000
mean      34.242138
std       29.681822
min        0.000000
25%       10.000000
50%       26.000000
75%       54.000000
max      153.000000
Name: Runs, dtype: float64
```

In [20]:

```
p_25 = dravid["Runs"].quantile(0.25) # Q1 or p_25
p_50 = dravid["Runs"].quantile(0.5)  # Q2 or p_50 or median
p_75 = dravid["Runs"].quantile(0.75) # Q3 or p_75
print(p_25, p_50, p_75)
```

```
10.0 26.0 54.0
```

In [21]:

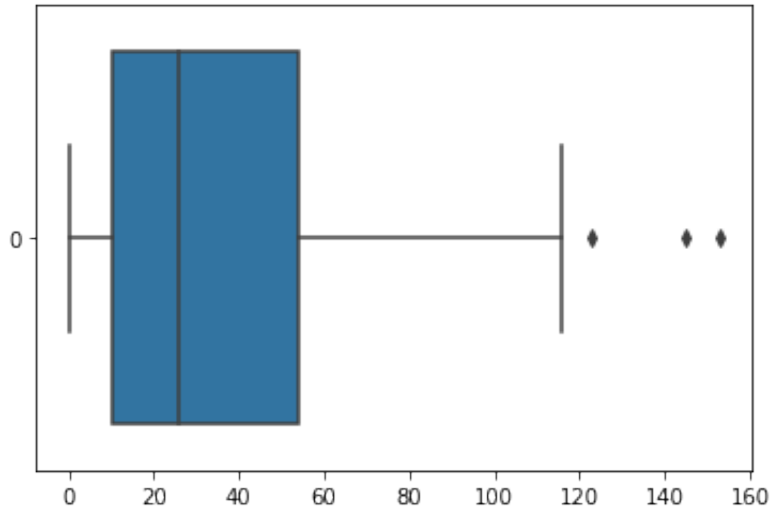
```
iqr = p_75 - p_25
lower = max(p_25 - 1.5*iqr, 0)
upper = p_75 + 1.5*iqr
print(lower, upper)
print(iqr)
```

```
0 120.0
44.0
```

In [22]:

```
sns.boxplot(data=dravid["Runs"], orient="h")
```

```
Out[22]: <AxesSubplot:>
```



```
In [23]: dravid_outlier = dravid[dravid["Runs"] > upper]
         len(dravid_outlier)

Out[23]: 3

In [24]: 3/318

Out[24]: 0.009433962264150943

In [25]: data = pd.read_html("https://stats.espncricinfo.com/ci/engine/player/25380

In [26]: type(data)

Out[26]: list

In [27]: len(data)

Out[27]: 7

In [38]: kohli = data[3]

In [39]: kohli
```

Out[39]:

| | Runs | Mins | BF | 4s | 6s | SR | Pos | Dismissal | Inns | Unnamed: 9 | Opposition | Gro |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 4 | 13 | 10 | 1 | 0 | 40.00 | 5 | caught | 1 | NaN | v West Indies | King |
| **1** | 15 | 72 | 54 | 2 | 0 | 27.77 | 5 | caught | 3 | NaN | v West Indies | King |
| **2** | 0 | 1 | 2 | 0 | 0 | 0.00 | 5 | caught | 1 | NaN | v West Indies | Bridge |
| **3** | 27 | 118 | 107 | 1 | 1 | 25.23 | 5 | caught | 3 | NaN | v West Indies | Bridge |
| **4** | 30 | 62 | 53 | 2 | 0 | 56.60 | 5 | caught | 2 | NaN | v West Indies | Ro |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **185** | 44 | 127 | 84 | 4 | 0 | 52.38 | 4 | lbw | 2 | NaN | v Australia | I |
| **186** | 20 | 47 | 31 | 3 | 0 | 64.51 | 4 | stumped | 4 | NaN | v Australia | I |
| **187** | 22 | 64 | 52 | 2 | 0 | 42.30 | 4 | lbw | 1 | NaN | v Australia | In |
| **188** | 13 | 35 | 26 | 2 | 0 | 50.00 | 4 | lbw | 3 | NaN | v Australia | In |
| **189** | 186 | 517 | 364 | 15 | 0 | 51.09 | 4 | caught | 2 | NaN | v Australia | Ahmed |

190 rows × 14 columns

# Height

In [41]:

```
df_hw = pd.read_csv("weight-height.csv")
df_hw
```

Out[41]:

|  | Gender | Height | Weight |
|---|---|---|---|
| **0** | Male | 73.847017 | 241.893563 |
| **1** | Male | 68.781904 | 162.310473 |
| **2** | Male | 74.110105 | 212.740856 |
| **3** | Male | 71.730978 | 220.042470 |
| **4** | Male | 69.881796 | 206.349801 |
| **...** | ... | ... | ... |
| **9995** | Female | 66.172652 | 136.777454 |
| **9996** | Female | 67.067155 | 170.867906 |
| **9997** | Female | 63.867992 | 128.475319 |
| **9998** | Female | 69.034243 | 163.852461 |
| **9999** | Female | 61.944246 | 113.649103 |

10000 rows × 3 columns

In [42]:

```
df_hw["Height"].describe()
```

Out[42]:

```
count    10000.000000
mean        66.367560
std          3.847528
min         54.263133
25%         63.505620
50%         66.318070
75%         69.174262
max         78.998742
Name: Height, dtype: float64
```

In [44]:

```
df_height = df_hw["Height"]
```

In [45]:

```
min_height = df_height.min()
min_height
```

Out[45]:

```
54.2631333250971
```

In [46]:

```
max_height = df_height.max()
max_height
```

Out[46]:

```
78.9987423463896
```

In [47]:

```
total = len(df_height)
```

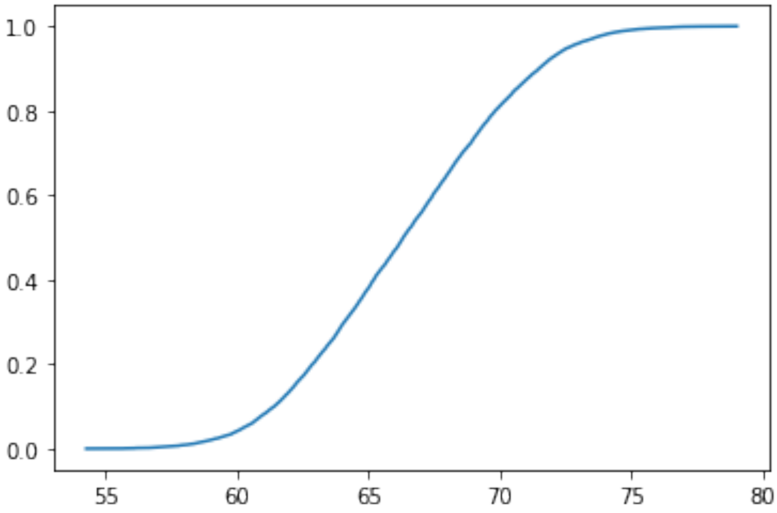
In [56]:

```
x_values = np.linspace(min_height, max_height, 1000)
y_values = []

for x in x_values:
    people_shorter_than_x = df_height[df_height <= x]
    num_people_shorter_than_x = len(people_shorter_than_x)
    frac_people_shorter_than_x = num_people_shorter_than_x / total
    y_values.append(frac_people_shorter_than_x)
plt.plot(x_values, y_values)
# e = ECDF(df_height)
# plt.plot(e.x, e.y, c="r")
```

Out[56]:

```
[<matplotlib.lines.Line2D at 0x28598b670>]
```



In [51]:

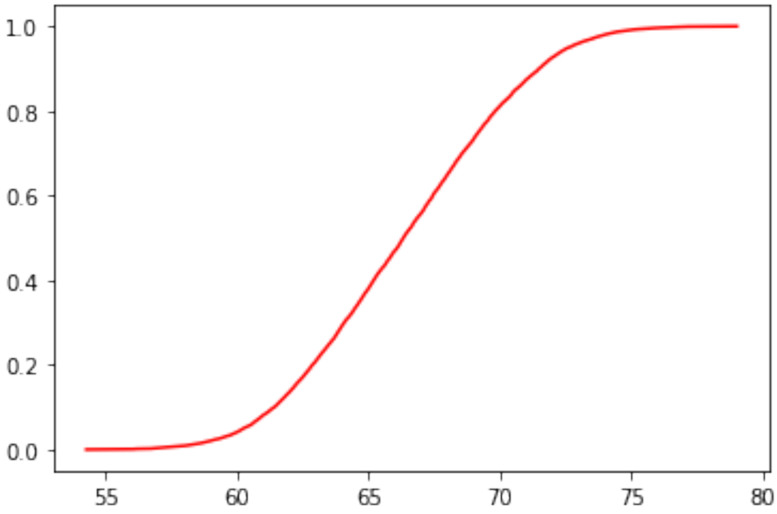
```
e = ECDF(df_height) # Empirical Cumulative Distribution Function (Empirica
```

In [52]:

```
plt.plot(e.x, e.y, c="r")
```

Out[52]:

```
[<matplotlib.lines.Line2D at 0x2856cca00>]
```



CDF: F(x) = P(Height <= x)

In [49]:

```
df_height.describe()
```

Out[49]:

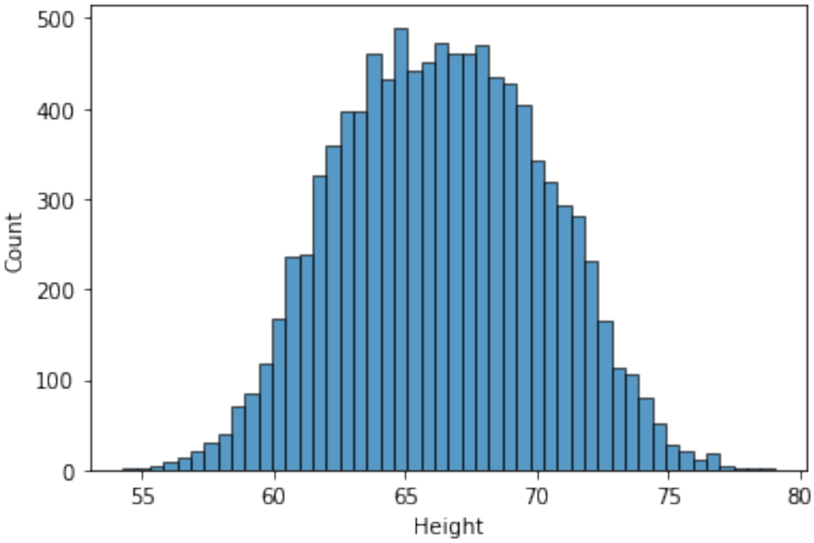
```
count    10000.000000
mean        66.367560
std          3.847528
min         54.263133
25%         63.505620
50%         66.318070
75%         69.174262
max         78.998742
Name: Height, dtype: float64
```

In [57]:

```
sns.histplot(df_height)
```

Out[57]:

```
<AxesSubplot:xlabel='Height', ylabel='Count'>
```


Count
Height