K-Means

Importing the libraries

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

Import the dataset

```
df = pd.read_csv("Cricket.csv", encoding='latin1')
```

Basic EDA

```
df.head()
```

```
                        Player       Span  Mat  Inns  NO   Runs    HS  
Ave  \
0         SR Tendulkar (INDIA)  1989-2012  463   452  41  18426  200*  
44.83   
1  KC Sangakkara (Asia/ICC/SL)  2000-2015  404   380  41  14234   169  
41.98   
2         RT Ponting (AUS/ICC)  1995-2012  375   365  39  13704   164  
42.03   
3      ST Jayasuriya (Asia/SL)  1989-2011  445   433  18  13430   189  
32.36   
4   DPMD Jayawardene (Asia/SL)  1998-2015  448   418  39  12650   144  
33.37   

      BF     SR  100  50   0  
0  21367  86.23   49  96  20  
1  18048  78.86   25  93  15  
2  17046  80.39   30  82  20  
3  14725  91.20   28  68  34  
4  16020  78.96   19  77  28  
```

```
df.shape
```

```
(79, 13)
```

```
df.describe()
```

```
              Mat        Inns         NO         Runs        Ave  \
count   79.000000   79.000000  79.000000    79.000000  79.000000   
mean   245.075949  230.544304  30.037975  7618.139241  38.523291   
std     74.211716   70.321022  14.421710  2551.873313   5.919093   
min    128.000000  127.000000   4.000000  5080.000000  23.570000   
25%    188.000000  177.000000  17.500000  5759.000000  34.600000   
50%    232.000000  217.000000  29.000000  6798.000000  37.870000   
75%    281.500000  261.500000  40.000000  8739.500000  41.845000   
```

```
max    463.000000  452.000000  70.000000  18426.000000  53.940000

                 BF          SR        100          50          0
count     79.000000   79.000000  79.000000  79.000000  79.000000
mean    9684.455696   79.295316  11.556962  46.443038  13.253165
std     3193.835825    9.925307   8.092014  16.351701   5.925755
min     5504.000000   60.570000   0.000000  23.000000   3.000000
25%     7393.500000   73.725000   6.000000  34.500000   9.000000
50%     9134.000000   77.730000  10.000000  42.000000  13.000000
75%    10976.000000   85.180000  15.500000  54.000000  16.000000
max    21367.000000  117.000000  49.000000  96.000000  34.000000
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 79 entries, 0 to 78
Data columns (total 13 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Player  79 non-null     object 
 1   Span    79 non-null     object 
 2   Mat     79 non-null     int64  
 3   Inns    79 non-null     int64  
 4   NO      79 non-null     int64  
 5   Runs    79 non-null     int64  
 6   HS      79 non-null     object 
 7   Ave     79 non-null     float64
 8   BF      79 non-null     int64  
 9   SR      79 non-null     float64
 10  100     79 non-null     int64  
 11  50      79 non-null     int64  
 12  0       79 non-null     int64  
dtypes: float64(2), int64(8), object(3)
memory usage: 8.1+ KB
```

Null Values

```
df.isnull().sum()
```

```
Player    0
Span      0
Mat       0
Inns      0
NO        0
Runs      0
HS        0
Ave       0
BF        0
SR        0
100       0
50        0
```

```
0         0
dtype: int64
```

Dealing the span column

```
# to convert the span into years of experience
# we first split the span into start and end and store in new column

df[['Strt','End']] = df.Span.str.split("-",expand=True)

# convert them as int and
# find years of exp as end year - start year
# later drop the unneceassry columns such as start, end and span
# we are only left with the experience column now

df[['Strt','End']]=df[['Strt','End']].astype(int)
df['Exp']=df['End']-df['Strt']
df=df.drop(['Strt','End','Span'], axis = 1)
df.head()
```

```
                        Player  Mat  Inns  NO   Runs    HS    Ave
BF  \
0         SR Tendulkar (INDIA)  463   452  41  18426  200*  44.83
21367
1  KC Sangakkara (Asia/ICC/SL)  404   380  41  14234   169  41.98
18048
2         RT Ponting (AUS/ICC)  375   365  39  13704   164  42.03
17046
3      ST Jayasuriya (Asia/SL)  445   433  18  13430   189  32.36
14725
4   DPMD Jayawardene (Asia/SL)  448   418  39  12650   144  33.37
16020

      SR  100  50   0  Exp
0  86.23   49  96  20   23
1  78.86   25  93  15   15
2  80.39   30  82  20   17
3  91.20   28  68  34   22
4  78.96   19  77  28   17
```

Dealing the HS column

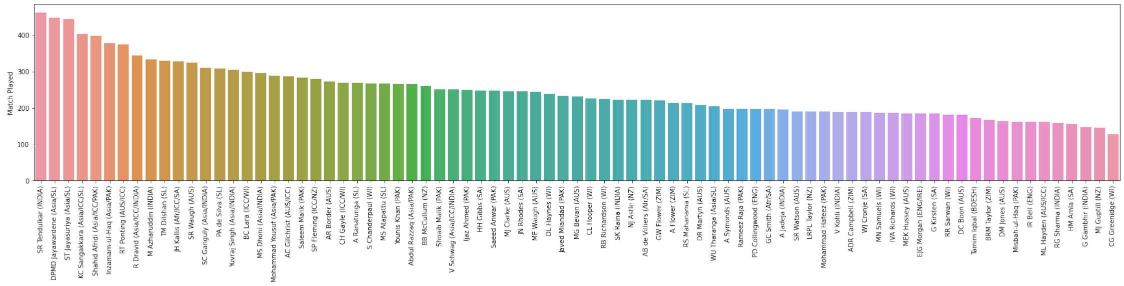
\d+ will extract the numeric value out the string

```
df.HS=df.HS.str.extract('(\d+)')
df.HS=df.HS.astype(int)

df.info()
```

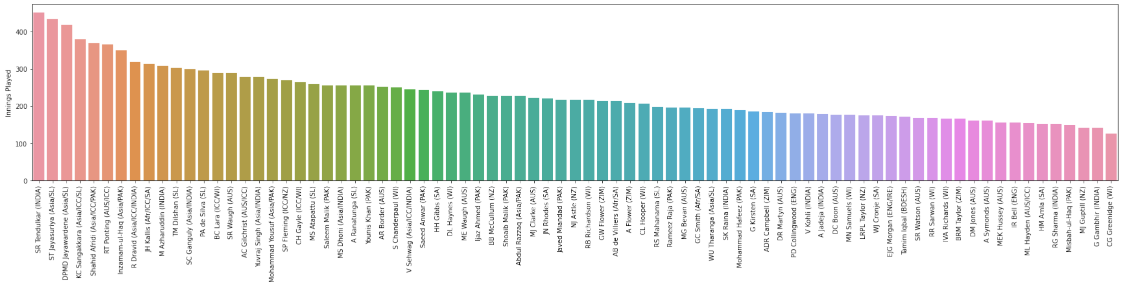
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 79 entries, 0 to 78
Data columns (total 13 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Player  79 non-null     object 
 1   Mat     79 non-null     int64  
 2   Inns    79 non-null     int64  
 3   NO      79 non-null     int64  
 4   Runs    79 non-null     int64  
 5   HS      79 non-null     int64  
 6   Ave     79 non-null     float64
 7   BF      79 non-null     int64  
 8   SR      79 non-null     float64
 9   100     79 non-null     int64  
 10  50      79 non-null     int64  
 11  0       79 non-null     int64  
 12  Exp     79 non-null     int64  
dtypes: float64(2), int64(10), object(1)
memory usage: 8.1+ KB
```

Data Visualization

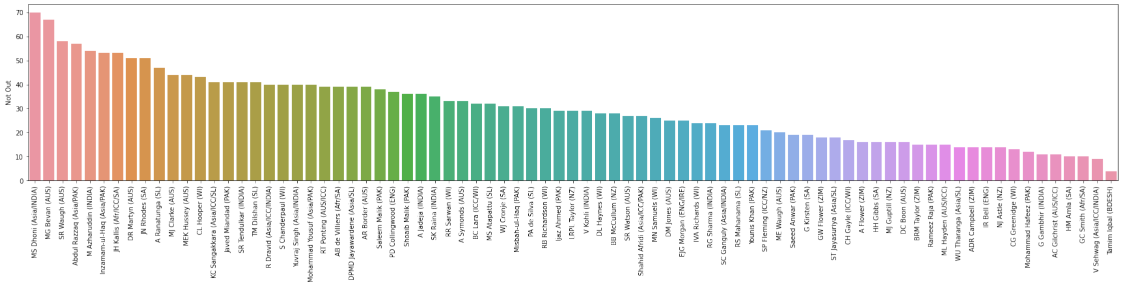
```
#Match Played

plt.figure(figsize = (30,5))
mat = df[['Player','Mat']].sort_values('Mat', ascending = False)
ax = sns.barplot(x='Player', y='Mat', data= mat)
ax.set(xlabel = '', ylabel= 'Match Played')
plt.xticks(rotation=90)
plt.show()
```



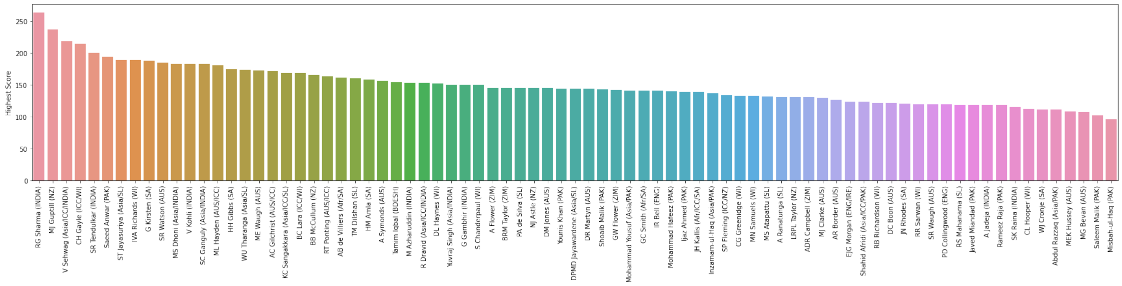
```
plt.figure(figsize = (30,5))
inns = df[['Player','Inns']].sort_values('Inns', ascending = False)
ax = sns.barplot(x='Player', y='Inns', data= inns)
ax.set(xlabel = '', ylabel= 'Innings Played')
plt.xticks(rotation=90)
plt.show()
```



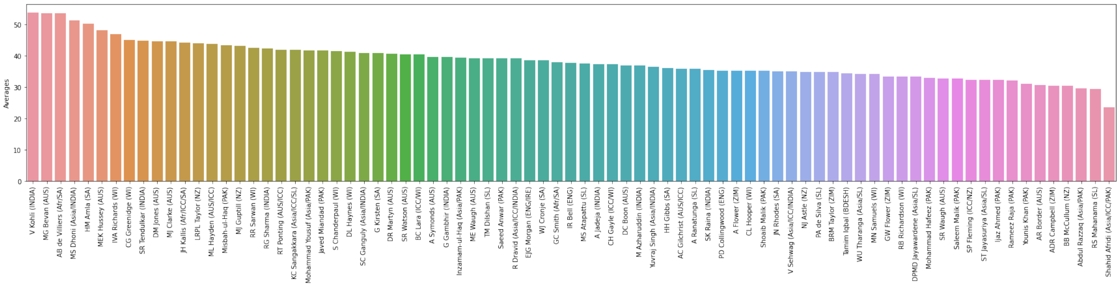
```
plt.figure(figsize = (30,5))
no = df[['Player','NO']].sort_values('NO', ascending = False)
ax = sns.barplot(x='Player', y='NO', data= no)
ax.set(xlabel = '', ylabel= 'Not Out')
plt.xticks(rotation=90)
plt.show()
```



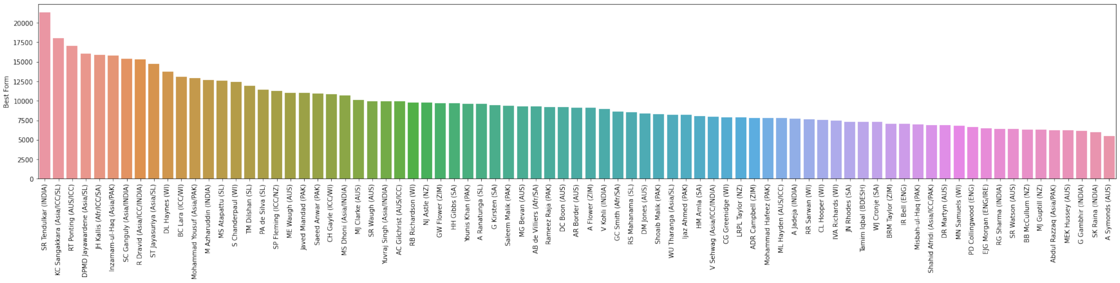
```
plt.figure(figsize = (30,5))
hs = df[['Player','HS']].sort_values('HS', ascending = False)
ax = sns.barplot(x='Player', y='HS', data= hs)
ax.set(xlabel = '', ylabel= 'Highest Score')
plt.xticks(rotation=90)
plt.show()
```



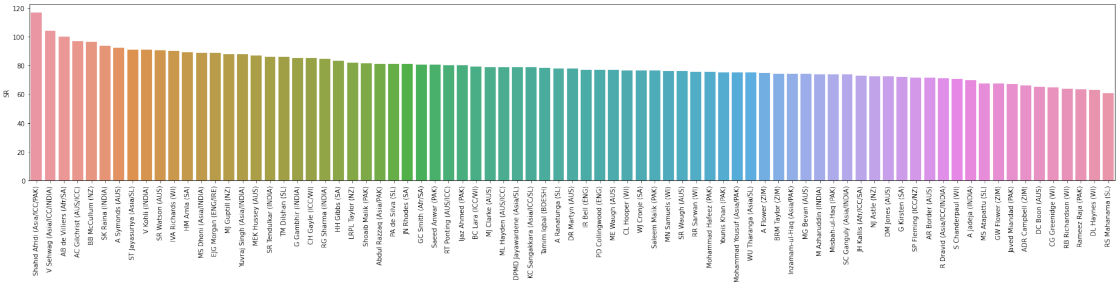
```
plt.figure(figsize = (30,5))
ave = df[['Player','Ave']].sort_values('Ave', ascending = False)
ax = sns.barplot(x='Player', y='Ave', data= ave)
ax.set(xlabel = '', ylabel= 'Averages')
plt.xticks(rotation=90)
plt.show()
```



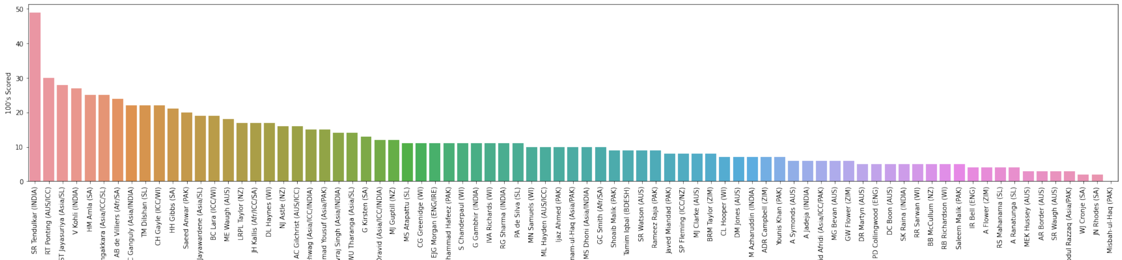
```
plt.figure(figsize = (30,5))
bf = df[['Player','BF']].sort_values('BF', ascending = False)
ax = sns.barplot(x='Player', y='BF', data= bf)
ax.set(xlabel = '', ylabel= 'Best Form')
plt.xticks(rotation=90)
plt.show()
```



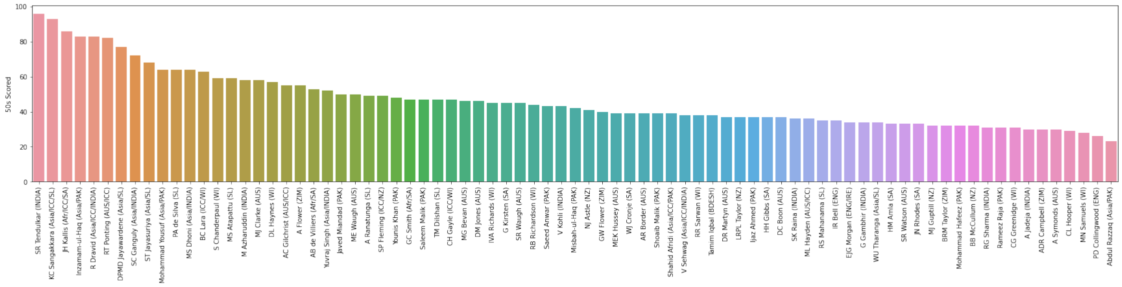
```
plt.figure(figsize = (30,5))
sr = df[['Player','SR']].sort_values('SR', ascending = False)
ax = sns.barplot(x='Player', y='SR', data= sr)
ax.set(xlabel = '', ylabel= 'SR')
plt.xticks(rotation=90)
plt.show()
```



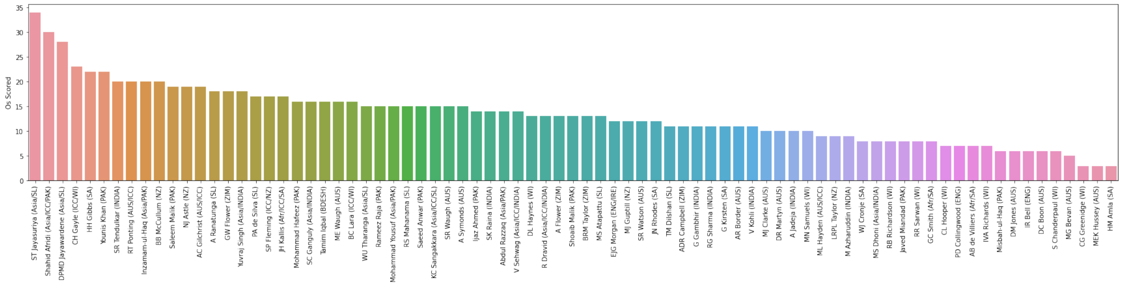
```
plt.figure(figsize = (30,5))
r100 = df[['Player','100']].sort_values('100', ascending = False)
ax = sns.barplot(x='Player', y='100', data= r100)
ax.set(xlabel = '', ylabel= "100's Scored" )
plt.xticks(rotation=90)
plt.show()
```



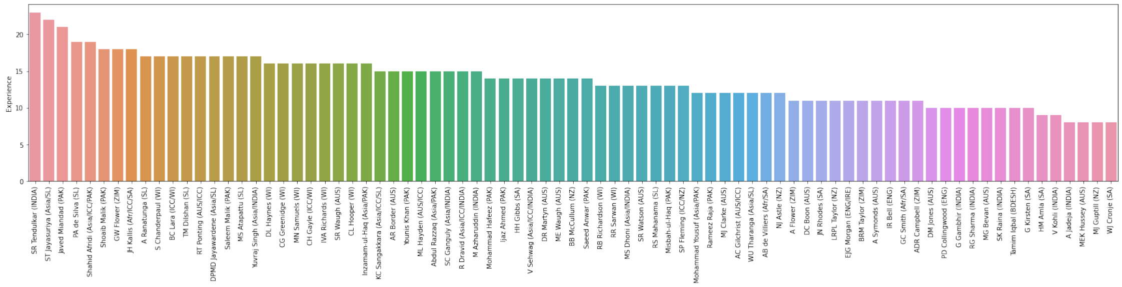
```
plt.figure(figsize = (30,5))
r50 = df[['Player','50']].sort_values('50', ascending = False)
ax = sns.barplot(x='Player', y='50', data= r50)
ax.set(xlabel = '', ylabel= "50s Scored")
plt.xticks(rotation=90)
plt.show()
```



```
plt.figure(figsize = (30,5))
r0 = df[['Player','0']].sort_values('0', ascending = False)
ax = sns.barplot(x='Player', y='0', data= r0)
ax.set(xlabel = '', ylabel= "0s Scored")
plt.xticks(rotation=90)
plt.show()
```

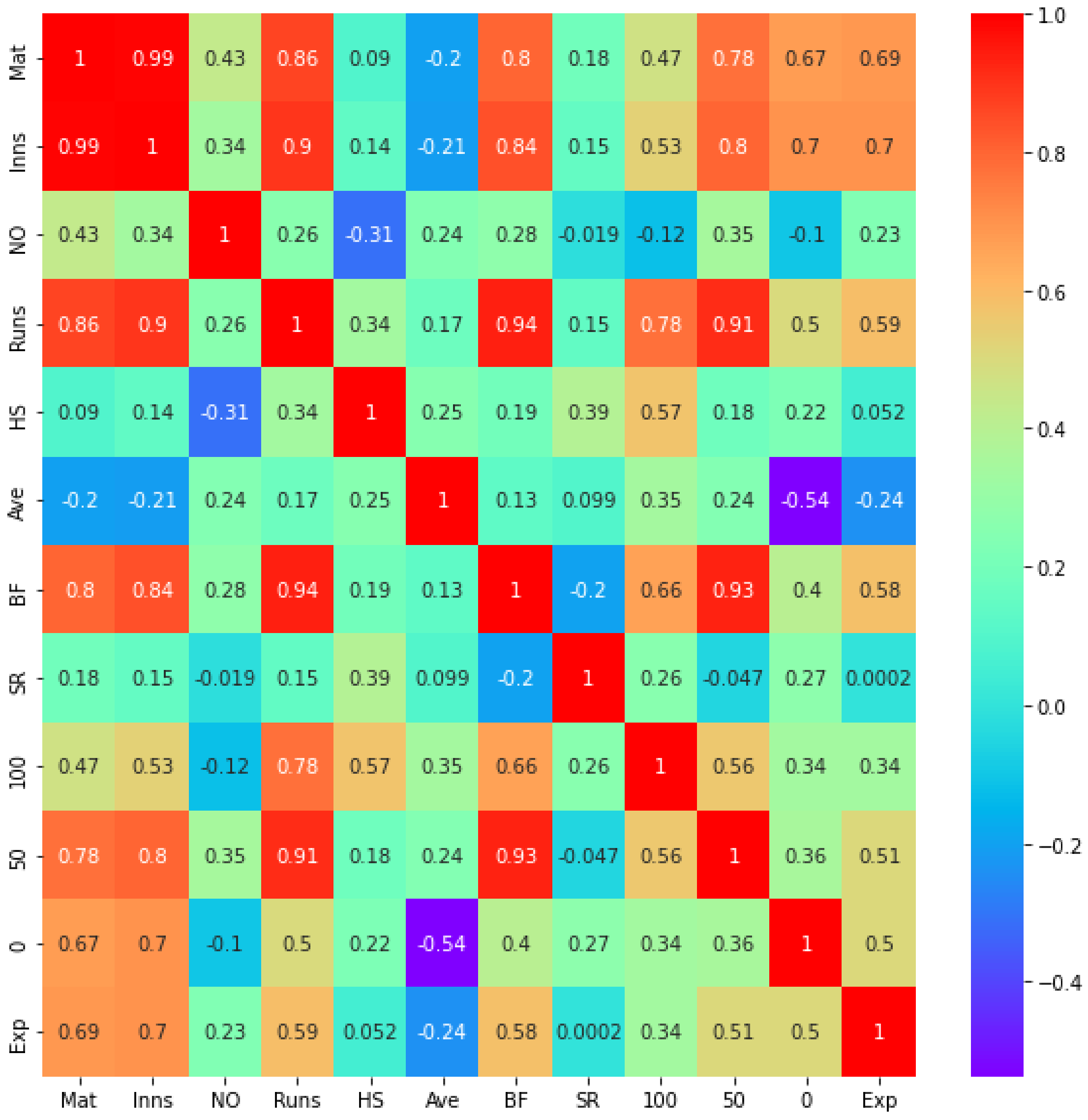


```
plt.figure(figsize = (30,5))
exp = df[['Player','Exp']].sort_values('Exp', ascending = False)
ax = sns.barplot(x='Player', y='Exp', data= exp)
ax.set(xlabel = '', ylabel= 'Experience')
plt.xticks(rotation=90)
plt.show()
```



Correlation heatmap

```
# Let's check the correlation coefficients to see which variables are
highly correlated

plt.figure(figsize = (10, 10))
sns.heatmap(df.corr(), annot = True, cmap="rainbow")
plt.savefig('Correlation')
plt.show()
```



Copying the original dataset

```
# Dropping Player field as final dataframe will only contain data
columns

df_drop = df.copy()
player = df_drop.pop('Player')

df_drop.head()
```

```
   Mat  Inns  NO   Runs   HS    Ave     BF     SR  100  50   0  Exp
0  463   452  41  18426  200  44.83  21367  86.23   49  96  20   23
1  404   380  41  14234  169  41.98  18048  78.86   25  93  15   15
2  375   365  39  13704  164  42.03  17046  80.39   30  82  20   17
3  445   433  18  13430  189  32.36  14725  91.20   28  68  34   22
4  448   418  39  12650  144  33.37  16020  78.96   19  77  28   17
```

Implementing the standardscaler

```
import sklearn
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
df_scaled = scaler.fit_transform(df_drop)
df_scaled
```

```
array([[ 2.95528204e+00,  3.16933340e+00,  7.64962749e-01,
         4.26232808e+00,  1.63244320e+00,  1.07229395e+00,
         3.68121424e+00,  7.03151526e-01,  4.65672622e+00,
         3.05005720e+00,  1.14583653e+00,  2.81278702e+00],
       [ 2.15517925e+00,  2.13891509e+00,  7.64962749e-01,
         2.60911662e+00,  6.35223595e-01,  5.87724608e-01,
         2.63538469e+00, -4.41394951e-02,  1.67188751e+00,
         2.86541772e+00,  2.96670622e-01,  3.80938324e-01],
       [ 1.76190839e+00,  1.92424461e+00,  6.25396797e-01,
         2.40009894e+00,  4.74381724e-01,  5.96225824e-01,
         2.31965067e+00,  1.10996904e-01,  2.29372891e+00,
         2.18840630e+00,  1.14583653e+00,  9.88900497e-01],
       [ 2.71118288e+00,  2.89741746e+00, -8.40045698e-01,
         2.29204075e+00,  1.27859108e+00, -1.04790945e+00,
         1.58829472e+00,  1.20709133e+00,  2.04499235e+00,
         1.32675540e+00,  3.52350108e+00,  2.50880593e+00],
       [ 2.75186607e+00,  2.68274698e+00,  6.25396797e-01,
         1.98442984e+00, -1.68985764e-01, -8.76184872e-01,
         1.99635416e+00, -3.39998611e-02,  9.25677829e-01,
         1.88067384e+00,  2.50450199e+00,  9.88900497e-01],
       [ 1.80259158e+00,  1.70957413e+00,  1.60235846e+00,
         1.62515607e+00, -3.94164385e-01,  1.69464756e-01,
         1.93081256e+00, -5.12590583e-01, -1.93636689e-01,
         2.24995279e+00,  1.14583653e+00,  6.84919410e-01],
       [ 1.12453837e+00,  1.19436497e+00,  1.60235846e+00,
         1.56205640e+00, -3.29827636e-01,  9.92382513e-01,
         1.95381514e+00, -6.49475641e-01,  6.76941269e-01,
         2.43459227e+00,  6.36336987e-01,  1.29288158e+00],
       [ 8.94000281e-01,  9.94005852e-01, -4.91130818e-01,
         1.47687184e+00,  1.08558084e+00,  4.24501251e-01,
         1.80603146e+00, -5.67344606e-01,  1.29878267e+00,
         1.57294137e+00,  4.66503804e-01,  3.80938324e-01],
       [ 1.34151540e+00,  1.25161043e+00,  6.95179773e-01,
         1.28993905e+00,  1.20529605e-01,  1.08255997e-01,
         1.76443775e+00, -8.16779601e-01,  5.50998708e-02,
         2.24995279e+00, -4.29957423e-02,  3.80938324e-01],
       [ 7.31267511e-01,  8.36580832e-01,  1.36915966e-01,
         1.09906254e+00,  6.35223595e-01,  3.32688113e-01,
         1.07183957e+00,  2.17681255e-02,  9.25677829e-01,
         1.01902294e+00,  4.66503804e-01,  9.88900497e-01],
       [ 1.15166050e+00,  1.03693995e+00,  7.64962749e-01,
         1.05370965e+00,  3.77876600e-01,  1.26958674e-01,
         7.08524880e-01,  7.03151526e-01,  1.29878267e+00,
         3.42790592e-02, -3.82662107e-01,  9.88900497e-01],
       [ 5.82095805e-01,  6.07598985e-01,  6.95179773e-01,
         8.28917057e-01, -2.65490887e-01,  5.41818038e-01,
         1.02646463e+00, -4.25389731e-01,  4.28204710e-01,
         1.08056943e+00,  2.96670622e-01, -5.31004936e-01],
       [ 5.68534741e-01,  6.93467177e-01, -1.32852653e+00,
         7.89085388e-01,  7.31728719e-01, -4.47723561e-01,
         7.48511155e-02,  1.78910632e+00,  5.52572990e-01,
         5.26651000e-01,  9.76003351e-01, -5.31004936e-01],
       [ 6.77023254e-01,  3.49994406e-01,  2.78866905e+00,
         7.40577513e-01,  1.08558084e+00,  2.17575185e+00,
         3.21892504e-01,  9.52586521e-01, -1.93636689e-01,
         1.08056943e+00, -8.92161654e-01, -2.27023850e-01],
       [ 1.20590476e+00,  1.10849678e+00,  1.67214144e+00,
         6.94041503e-01,  1.20529605e-01, -2.72598501e-01,
         9.40441285e-01, -5.34897778e-01, -5.66741528e-01,
         7.11290478e-01, -7.22328471e-01,  3.80938324e-01],
       [-3.12934430e-01, -2.51082944e-01,  6.25396797e-01,
         6.70773499e-01,  4.10044975e-01,  2.55490610e+00,
        -1.22718974e-01,  2.12472820e+00,  1.54751923e+00,
         4.03558015e-01, -1.06199484e+00, -5.31004936e-01],
       [ 8.53317089e-01,  9.36760390e-01, -2.64998643e-03,
         6.56970445e-01, -1.36817390e-01, -6.16047648e-01,
         5.54124013e-01,  1.86030195e-01, -6.92684089e-02,
         1.08056943e+00,  6.36336987e-01,  1.59686267e+00],
       [ 3.24435586e-01,  4.78796695e-01, -9.09828674e-01,
         6.32124948e-01,  2.11496882e+00, -2.02888526e-01,
         3.62225791e-01,  5.89587626e-01,  1.29878267e+00,
         3.42790592e-02,  1.65533608e+00,  6.84919410e-01],
       [ 2.60921741e-02,  1.92569386e-01, -7.70262722e-01,
         4.75558882e-01,  1.43943296e+00,  1.16757214e-01,
         3.94996588e-01,  1.39387879e-01,  1.05004611e+00,
        -2.11906911e-01,  2.96670622e-01,  7.69572371e-02],
       [ 3.10874522e-01,  2.92748944e-01,  6.95179773e-01,
         4.57417725e-01,  2.40244821e-02,  5.23115362e-01,
         8.58199191e-01, -8.67477771e-01, -6.92684089e-02,
         7.72836970e-01, -1.23182802e+00,  9.88900497e-01],
       [ 7.99072832e-01,  6.79155812e-01,  6.95179773e-01,
         4.27051007e-01,  2.40244821e-02, -3.35507504e-01,
         7.54813231e-02,  8.49162255e-01,  3.03836430e-01,
         3.42011522e-01,  8.06170169e-01,  9.88900497e-01],
       [-9.59574035e-02,  9.23898277e-02, -1.42215938e-01,
         4.06149240e-01,  8.83612309e-02,  4.84009766e-01,
         1.26751904e+00, -1.64315977e+00,  6.76941269e-01,
         6.49743985e-01, -4.29957423e-02,  6.84919410e-01],
       [ 3.10874522e-01,  4.07239868e-01,  1.36915966e-01,
         3.59218857e-01, -5.55006257e-01, -1.62082687e-01,
         9.16808500e-01, -1.17369472e+00, -6.92684089e-02,
         7.72836970e-01, -4.29957423e-02,  9.88900497e-01],
       [-1.45910184e-02,  7.80784622e-02, -7.00479746e-01,
         3.47782041e-01,  7.63897093e-01,  1.40560620e-01,
         4.31233526e-01, -2.42876320e-01,  8.01309549e-01,
         2.18918537e-01,  4.66503804e-01,  7.69572371e-02],
       [ 8.03364308e-02,  2.06880751e-01, -1.46809248e+00,
         2.58259379e-01,  2.24364231e+00, -5.90543998e-01,
        -5.53150781e-01,  2.53842527e+00,  4.28204710e-01,
        -5.19639374e-01,  1.26837440e-01,  7.69572371e-02],
       [-7.74010612e-01, -7.23358004e-01, -7.24329624e-02,
         2.08174012e-01,  1.08558084e+00,  2.62121559e+00,
        -2.30799582e-01,  1.18579810e+00,  1.92062407e+00,
        -2.11906911e-01, -3.82662107e-01, -1.44294820e+00],
       [ 3.96532383e-02,  1.35323924e-01, -9.79611650e-01,
         1.87666618e-01,  8.28233842e-01, -4.06917722e-01,
         1.15152494e-02,  4.02004398e-01,  1.17441439e+00,
        -5.81185867e-01,  1.48550290e+00,  7.69572371e-02],
       [ 2.07381286e+00,  1.98149007e+00, -2.11998914e-01,
         1.75835429e-01, -8.12353252e-01, -2.54242330e+00,
        -8.79913433e-01,  3.82311689e+00, -6.91109807e-01,
        -4.58092882e-01,  2.84416836e+00,  1.59686267e+00],
       [ 4.73607292e-01,  5.50353523e-01, -6.30696770e-01,
         1.65187359e-01, -4.90669508e-01, -1.04110847e+00,
         4.90788147e-01, -7.91430516e-01, -4.42373248e-01,
         1.57372044e-01,  6.36336987e-01, -2.27023850e-01],
       [-1.02995424e-03, -1.07969289e-01,  9.74311677e-01,
         1.43102473e-01, -6.19343006e-01,  1.02978787e+00,
         1.32200009e-01, -3.19719343e-02, -4.42373248e-01,
         7.11290478e-01, -5.52495289e-01, -5.31004936e-01],
       [ 1.08385518e+00,  8.22269467e-01,  1.95127334e+00,
        -1.93791879e-02, -9.41026749e-01, -9.56096307e-01,
         9.02912023e-02, -3.43258696e-01, -1.06421465e+00,
        -8.88139260e-02,  2.96670622e-01,  6.84919410e-01],
       [ 3.24435586e-01,  3.49994406e-01,  1.18366061e+00,
        -6.39433328e-02, -5.87174631e-01, -4.56224778e-01,
        -3.57503223e-02, -1.41479981e-01, -9.39846367e-01,
         1.57372044e-01,  8.06170169e-01,  9.88900497e-01],
       [-1.63762724e-01, -1.79526116e-01,  7.64962749e-01,
        -9.35213051e-02, -9.73195124e-01,  5.40117795e-01,
         4.18944478e-01, -1.24568612e+00, -4.42373248e-01,
         2.18918537e-01, -8.92161654e-01,  2.20482484e+00],
       [ 2.70191329e-01,  3.49994406e-01, -4.91130818e-01,
        -1.45578536e-01, -1.68985764e-01, -1.23833670e+00,
        -1.77894050e-02, -4.06124427e-01, -5.66741528e-01,
         9.58255518e-02,  1.48550290e+00,  3.80938324e-01],
       [-1.20796467e+00, -1.10976487e+00, -1.39830951e+00,
        -1.70424033e-01,  3.13539852e-01,  1.99382582e+00,
        -5.09036247e-01,  9.89089203e-01,  1.67188751e+00,
        -8.27371837e-01, -1.74132756e+00, -1.44294820e+00],
       [ 5.14290484e-01,  3.64305772e-01,  5.55613821e-01,
        -1.76734000e-01, -1.52005749e+00, -9.59496794e-01,
        -9.49898388e-02, -2.92560526e-01, -8.15478087e-01,
         3.42790592e-02,  9.76003351e-01,  9.88900497e-01],
       [-2.99373366e-01, -1.93837482e-01, -1.11917760e+00,
        -2.08283838e-01, -1.36817390e-01, -6.12647161e-01,
         2.38042981e-02, -6.74824726e-01,  5.52572990e-01,
        -3.34999896e-01,  9.76003351e-01, -5.31004936e-01],
       [-6.51961035e-01, -5.22998887e-01, -1.39830951e+00,
        -2.48115507e-01, -2.65490887e-01, -9.23727119e-02,
        -3.26591140e-01,  1.53583367e-01, -1.93636689e-01,
         3.42790592e-02, -8.92161654e-01, -8.34986023e-01],
       [-1.77323788e-01, -4.94376157e-01,  2.57932012e+00,
        -2.78482225e-01, -1.32704724e+00,  2.56000683e+00,
        -1.14841379e-01, -5.20702290e-01, -6.91109807e-01,
        -2.72674334e-02, -1.40166120e+00, -1.13896711e+00],
       [-8.14693805e-01, -6.51801177e-01, -7.70262722e-01,
        -3.23440743e-01,  1.24642271e+00,  4.12599548e-01,
        -7.82893368e-02, -7.35662529e-01,  1.79468150e-01,
        -8.88139260e-02, -3.82662107e-01, -1.13896711e+00],
       [-4.34984008e-01, -3.22639771e-01, -9.79611650e-01,
        -3.28173219e-01, -1.36817390e-01, -5.41236943e-01,
        -1.85109529e-01, -4.77101864e-01, -9.39846367e-01,
         5.26651000e-01, -4.29957423e-02, -8.34986023e-01],
       [ 9.38974950e-02, -3.64124615e-02,  4.16047869e-01,
        -3.36455051e-01, -2.01154139e-01, -5.59939619e-01,
        -4.43179550e-01,  2.45854035e-01, -3.18004968e-01,
        -4.58092882e-01, -4.29957423e-02,  1.29288158e+00],
       [-7.87571676e-01, -9.09405755e-01, -4.21347842e-01,
        -3.53807461e-01,  1.27859108e+00,  1.44124674e+00,
        -7.03770402e-01,  1.10569499e+00, -6.92684089e-02,
        -8.88139260e-02, -1.06199484e+00,  6.84919410e-01],
       [-3.26495494e-01, -2.36771578e-01, -8.40045698e-01,
        -4.12963406e-01, -2.33322513e-01, -8.50681223e-01,
         1.21454570e-02, -1.18789020e+00, -6.91109807e-01,
        -3.96546389e-01,  8.06170169e-01,  1.29288158e+00],
       [ 6.67753666e-02,  2.08330004e-02, -7.24329624e-02,
        -4.15724016e-01, -3.29827636e-01, -1.05301018e+00,
        -4.75950347e-01,  1.01871234e-01, -1.93636689e-01,
        -5.81185867e-01,  1.26837440e-01,  7.69572371e-02],
       [ 3.78679843e-01,  3.07060310e-01,  6.25396797e-01,
        -4.31498935e-01, -7.15848129e-01, -1.34375178e+00,
        -1.73450688e-01, -7.98528260e-01, -1.06421465e+00,
        -4.58092882e-01, -3.82662107e-01,  3.80938324e-01],
       [-7.46888484e-01, -7.80603466e-01, -7.24329624e-02,
        -4.46485108e-01, -5.87174631e-01,  9.51576673e-01,
        -5.60398168e-01,  2.77286901e-01,  6.76941269e-01,
        -5.81185867e-01, -7.22328471e-01, -8.34986023e-01],
       [-2.85812302e-01, -1.93837482e-01, -2.64998643e-03,
        -5.40345873e-01, -8.76690001e-01, -8.69383899e-01,
         3.67235543e-02, -1.57725215e+00, -8.15478087e-01,
        -1.50360419e-01, -8.92161654e-01, -2.27023850e-01],
       [-5.57033585e-01, -5.37310253e-01, -1.11917760e+00,
        -5.82938153e-01,  7.96065467e-01, -7.18062246e-01,
        -4.73114412e-01, -4.32487475e-01,  3.03836430e-01,
        -7.65825344e-01,  2.96670622e-01, -5.31004936e-01],
       [-1.14015934e+00, -1.08114214e+00, -1.04939463e+00,
        -5.85698764e-01,  1.02124409e+00,  8.97168888e-01,
        -6.04197598e-01, -3.39998611e-02, -1.93636689e-01,
        -6.42732359e-01, -7.22328471e-01,  3.80938324e-01],
       [ 2.02386008e-01, -3.64124615e-02, -1.42215938e-01,
        -6.05417412e-01,  5.38718472e-01, -1.37945689e+00,
        -1.06267364e+00,  1.73131041e+00, -8.15478087e-01,
        -8.88918330e-01,  1.14583653e+00,  7.69572371e-02],
       [-1.09947615e+00, -9.95273948e-01, -3.51564866e-01,
        -6.11333006e-01, -1.36817390e-01,  1.03488860e+00,
        -4.16710830e-01, -6.82936433e-01, -5.66741528e-01,
        -2.72674334e-02, -1.23182802e+00, -1.13896711e+00],
       [-8.68938061e-01, -7.66292100e-01, -9.79611650e-01,
        -6.52347795e-01, -8.76690001e-01, -2.52195582e-01,
        -1.66203300e-01, -1.43631123e+00, -8.15478087e-01,
        -5.81185867e-01, -1.23182802e+00, -8.34986023e-01],
       [-1.02995424e-03, -1.50903385e-01,  1.46279251e+00,
        -6.63784611e-01, -9.08858375e-01, -5.80342539e-01,
        -7.40007341e-01,  1.62709037e-01, -1.18858293e+00,
        -8.27371837e-01, -2.12828925e-01, -8.34986023e-01],
       [-7.46888484e-01, -5.80244349e-01, -1.25874355e+00,
        -6.87052616e-01, -2.97659262e-01, -9.37393631e-01,
        -6.01991871e-01, -3.76719488e-01, -6.92684089e-02,
        -8.88918330e-01,  4.66503804e-01,  7.69572371e-02],
       [-6.38399970e-01, -4.80064791e-01, -1.04939463e+00,
        -7.00855669e-01, -9.73195124e-01, -1.09381601e+00,
        -1.44461137e-01, -1.62085257e+00, -3.18004968e-01,
        -9.50464822e-01,  2.96670622e-01, -5.31004936e-01],
       [-8.68938061e-01, -8.80783024e-01,  2.06698941e-01,
        -7.15447469e-01, -9.41026749e-01,  7.05041395e-01,
        -6.36968394e-01, -3.60496074e-01, -8.15478087e-01,
        -5.19639374e-01, -8.92161654e-01, -2.27023850e-01],
       [-8.14693805e-01, -8.09226197e-01, -3.51564866e-01,
        -7.28067404e-01, -8.12353252e-01,  3.51455355e-02,
        -9.99337778e-01,  9.45488777e-01, -6.92684089e-02,
        -7.65825344e-01, -2.12828925e-01, -8.34986023e-01],
       [-2.45129109e-01, -3.51262502e-01,  9.04528701e-01,
        -7.32405506e-01, -1.16620537e+00, -5.41236943e-01,
        -6.82973551e-01, -2.70253332e-01, -5.66741528e-01,
        -1.07355781e+00, -1.06199484e+00,  6.84919410e-01],
       [-7.46888484e-01, -8.80783024e-01, -2.11998914e-01,
        -7.33982998e-01,  1.14991759e+00,  3.42889573e-01,
        -1.04597314e+00,  1.13003011e+00, -3.18004968e-01,
        -8.27371837e-01, -2.12828925e-01, -2.27023850e-01],
       [-9.77426575e-01, -8.52160293e-01, -1.81700736e+00,
        -7.39504220e-01,  1.52697980e-01, -7.04460299e-01,
        -7.40952652e-01, -9.99074817e-02, -3.18004968e-01,
        -5.19639374e-01,  4.66503804e-01, -1.13896711e+00],
       [-2.99373366e-01, -5.51621618e-01,  3.46264893e-01,
        -8.08519488e-01, -1.06970025e+00, -5.20834023e-01,
        -1.18052247e+00,  1.46666596e+00, -8.15478087e-01,
        -6.42732359e-01,  1.26837440e-01, -1.13896711e+00],
       [-7.74010612e-01, -7.94914831e-01,  6.71329895e-02,
        -8.09702607e-01, -1.19837374e+00,  1.98433458e-02,
        -7.58598466e-01, -2.86476746e-01, -1.18858293e+00,
        -4.58092882e-01, -8.92161654e-01, -1.74692928e+00],
       [-1.34357531e+00, -1.25287853e+00, -9.79611650e-01,
        -8.35336850e-01,  2.82267305e+00,  8.12156723e-01,
        -1.07622311e+00,  8.55246035e-01,  5.50998708e-02,
        -8.88918330e-01, -2.12828925e-01, -1.74692928e+00],
       [-8.14693805e-01, -1.05251941e+00,  9.74311677e-01,
        -8.58210482e-01, -1.29487887e+00,  1.63677472e+00,
        -1.08441581e+00,  7.97450122e-01, -1.06421465e+00,
        -4.58092882e-01, -1.74132756e+00, -1.74692928e+00],
       [-1.18084254e+00, -1.12407624e+00, -4.21347842e-01,
        -8.60971093e-01,  3.69121916e+00,  6.69336286e-01,
        -1.02643671e+00,  5.33819639e-01, -6.92684089e-02,
        -9.50464822e-01, -3.82662107e-01, -1.13896711e+00],
       [-1.14015934e+00, -1.05251941e+00, -1.11917760e+00,
        -8.68464179e-01, -2.65490887e-01, -1.11075388e-01,
        -8.39895249e-01, -2.16513272e-01, -9.39846367e-01,
        -7.04278852e-01, -1.23182802e+00, -8.34986023e-01],
       [-6.65522099e-01, -7.37669370e-01,  4.16047869e-01,
        -8.90943438e-01, -9.73195124e-01, -1.79085120e-01,
        -6.32556941e-01, -9.62790330e-01, -6.91109807e-01,
        -1.01201131e+00, -5.52495289e-01, -1.74692928e+00],
       [-5.02789329e-01, -6.94735273e-01,  1.46279251e+00,
        -8.96070286e-01, -1.68985764e-01,  3.87095898e-01,
        -8.84639990e-01, -1.58717359e-01, -8.15478087e-01,
        -5.81185867e-01, -5.52495289e-01,  7.69572371e-02],
       [-1.05879296e+00, -9.23717121e-01, -1.04939463e+00,
        -9.30775107e-01, -1.36817390e-01, -6.29649594e-01,
        -8.26975993e-01, -4.89269425e-01, -4.42373248e-01,
        -8.88918330e-01, -4.29957423e-02, -8.34986023e-01],
       [-1.33001424e+00, -1.25287853e+00, -1.32852653e+00,
        -9.38662566e-01,  2.40244821e-02,  1.96668649e-01,
        -1.11561108e+00,  6.03783113e-01, -6.92684089e-02,
        -7.65825344e-01, -3.82662107e-01, -1.13896711e+00],
       [-7.74010612e-01, -6.66112542e-01, -1.11917760e+00,
        -9.59564333e-01, -5.87174631e-01, -1.36415470e+00,
        -5.83085643e-01, -1.32984508e+00, -5.66741528e-01,
        -1.01201131e+00, -3.82662107e-01, -8.34986023e-01],
       [-7.87571676e-01, -7.66292100e-01, -2.81781890e-01,
        -9.61536198e-01, -5.22837882e-01, -7.18062246e-01,
        -9.10793607e-01, -3.09797904e-01, -1.93636689e-01,
        -1.13510430e+00, -5.52495289e-01,  6.84919410e-01],
       [-4.34984008e-01, -4.65753426e-01, -4.91130818e-01,
        -9.68634912e-01, -9.73195124e-01, -1.53587927e+00,
        -3.66609324e-01, -1.89867854e+00, -9.39846367e-01,
        -7.04278852e-01,  2.96670622e-01, -2.27023850e-01],
       [-1.58767446e+00, -1.48186037e+00, -1.18896058e+00,
        -9.79677355e-01, -5.22837882e-01,  1.10629881e+00,
        -5.59767961e-01, -1.45760447e+00, -6.92684089e-02,
        -9.50464822e-01, -1.74132756e+00,  6.84919410e-01],
       [-1.12659828e+00, -1.16701033e+00,  6.71329895e-02,
        -9.84409830e-01, -1.71306773e+00,  8.29159156e-01,
        -8.63212931e-01, -5.62274789e-01, -1.43731949e+00,
        -2.73453404e-01, -1.23182802e+00, -2.27023850e-01],
       [-6.51961035e-01, -7.09046639e-01,  4.85830845e-01,
        -9.96241019e-01, -9.41026749e-01, -5.37836456e-01,
        -9.67512293e-01, -2.34764613e-01, -8.15478087e-01,
        -1.25819729e+00, -1.06199484e+00, -1.13896711e+00],
       [-6.38399970e-01, -9.95273948e-01,  2.06698941e-01,
        -9.97818511e-01,  2.17034728e-01,  2.08570352e-01,
        -1.31727752e+00,  1.33282279e+00, -6.91109807e-01,
        -1.01201131e+00,  2.96670622e-01, -8.34986023e-01],
       [ 2.70191329e-01, -3.64124615e-02,  1.88149036e+00,
        -1.00097349e+00, -1.19837374e+00, -1.50017416e+00,
        -1.08157987e+00,  1.98197756e-01, -1.06421465e+00,
        -1.44283676e+00,  1.26837440e-01,  3.80938324e-01]])
```

```
df_df1 = pd.DataFrame(df_scaled, columns = [ 'Mat', 'Inns', 'NO',
'Runs', 'HS', 'Ave', 'BF', 'SR', '100',
                                             '50', '0', 'Exp'])
df_df1.head()
```

```
        Mat      Inns        NO      Runs        HS       Ave
BF  \
0  2.955282  3.169333  0.764963  4.262328  1.632443  1.072294
3.681214
1  2.155179  2.138915  0.764963  2.609117  0.635224  0.587725
2.635385
2  1.761908  1.924245  0.625397  2.400099  0.474382  0.596226
2.319651
3  2.711183  2.897417 -0.840046  2.292041  1.278591 -1.047909
1.588295
4  2.751866  2.682747  0.625397  1.984430 -0.168986 -0.876185
1.996354

         SR       100        50         0       Exp
0  0.703152  4.656726  3.050057  1.145837  2.812787
1 -0.044139  1.671888  2.865418  0.296671  0.380938
2  0.110997  2.293729  2.188406  1.145837  0.988900
3  1.207091  2.044992  1.326755  3.523501  2.508806
4 -0.034000  0.925678  1.880674  2.504502  0.988900
```

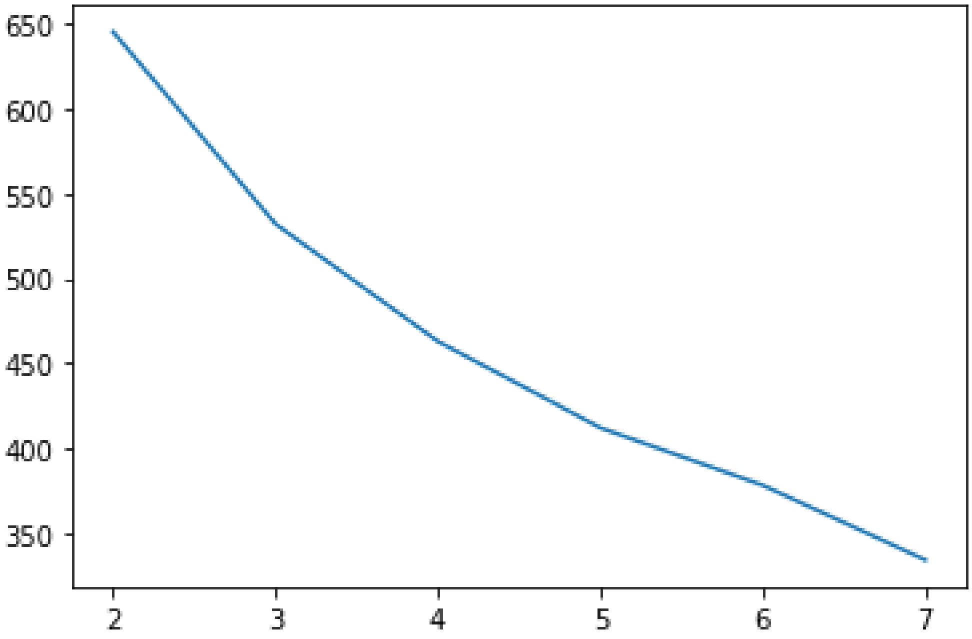
Implementing K-Means

```
from sklearn.cluster import KMeans
```

Elbow-Method

```
# Elbow curve method to find the ideal number of clusters.

clusters=list(range(2,8))
ssd = []
for num_clusters in clusters:
    model_clus = KMeans(n_clusters = num_clusters)
    model_clus.fit(df_df1)
    ssd.append(model_clus.inertia_)

plt.plot(clusters,ssd);
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
```



Model Building

```
cluster = KMeans(n_clusters=4)
cluster.fit(df_df1)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to
'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(

KMeans(n_clusters=4)
```

Figure out how many clusters

```
# Cluster labels

cluster.labels_
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 2, 0, 2, 0, 0, 0, 0, 0,
0,
       0, 0, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 3, 3, 2, 3, 0, 2,
0,
       0, 0, 3, 3, 3, 2, 0, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 2, 3,
2,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3], dtype=int32)
```

Cluster labels wrt dataset

```
df['Cluster_Id'] = cluster.labels_
df.head()
```

```
                        Player  Mat  Inns  NO   Runs   HS    Ave  
BF  \
0          SR Tendulkar (INDIA)  463   452  41  18426  200  44.83  
21367   
1  KC Sangakkara (Asia/ICC/SL)  404   380  41  14234  169  41.98  
18048   
2          RT Ponting (AUS/ICC)  375   365  39  13704  164  42.03  
17046   
3      ST Jayasuriya (Asia/SL)  445   433  18  13430  189  32.36  
14725   
4   DPMD Jayawardene (Asia/SL)  448   418  39  12650  144  33.37  
16020   

      SR  100  50   0  Exp  Cluster_Id  
0  86.23   49  96  20   23           1  
1  78.86   25  93  15   15           1  
2  80.39   30  82  20   17           1  
3  91.20   28  68  34   22           1  
4  78.96   19  77  28   17           1  
```

1st Cluster

```
df[df['Cluster_Id']==0].sort_values(by = ['NO','Ave','SR'], ascending
= [False,False,False]).head()
```

```
                  Player  Mat  Inns  NO  Runs   HS    Ave     BF  
SR  100  \
30         SR Waugh (AUS)  325   288  58  7569  120  32.90   9971  
75.91    3   
14  M Azharuddin (INDIA)  334   308  54  9378  153  36.92  12669  
74.02    7   
31      A Ranatunga (SL)  269   255  47  7456  131  35.84   9571  
77.90    4   
29       MJ Clarke (AUS)  245   223  44  7981  130  44.58  10104  
78.98    8   
32   Javed Miandad (PAK)  233   218  41  7381  119  41.70  11014  
67.01    8   

    50   0  Exp  Cluster_Id  
30  45  15   16           0  
14  58   9   15           0  
31  49  18   17           0  
29  58  10   12           0  
32  50   8   21           0  
```

2nd cluster

```
df[df['Cluster_Id']==1].sort_values(by = ['NO','Ave','SR'], ascending
= [False,False,False]).head()
```

```
                        Player  Mat  Inns  NO   Runs   HS    Ave  
BF  \
6       JH Kallis (Afr/ICC/SA)  328   314  53  11579  139  44.36
15885
5    Inzamam-ul-Haq (Asia/PAK)  378   350  53  11739  137  39.52
15812
0         SR Tendulkar (INDIA)  463   452  41  18426  200  44.83
21367
1  KC Sangakkara (Asia/ICC/SL)  404   380  41  14234  169  41.98
18048
8    R Dravid (Asia/ICC/INDIA)  344   318  40  10889  153  39.16
15284

      SR  100  50   0  Exp  Cluster_Id
6  72.89   17  86  17   18           1
5  74.24   10  83  20   16           1
0  86.23   49  96  20   23           1
1  78.86   25  93  15   15           1
8  71.24   12  83  13   15           1
```

3rd cluster

```
df[df['Cluster_Id']==2].sort_values(by = ['NO','Ave','SR'], ascending
= [False,False,False]).head()
```

```
                     Player  Mat  Inns  NO  Runs   HS    Ave     BF
SR  \
13    MS Dhoni (Asia/INDIA)  295   255  70  9496  183  51.32  10706
88.69
15  AB de Villiers (Afr/SA)  222   213  39  9319  162  53.55   9295
100.25
25          V Kohli (INDIA)  188   180  29  8146  183  53.94   8952
90.99
59          SR Watson (AUS)  190   169  27  5757  185  40.54   6365
90.44
42        IVA Richards (WI)  187   167  24  6721  189  47.00   7451
90.20

    100  50   0  Exp  Cluster_Id
13   10  64   8   13           2
15   24  53   7   12           2
25   27  43  11    9           2
59    9  33  12   13           2
42   11  45   7   16           2
```

4th cluster

```
df[df['Cluster_Id']==3].sort_values(by = ['NO','Ave','SR'], ascending
= [False,False,False]).head()
```

```
                     Player  Mat  Inns  NO  Runs   HS    Ave    BF
SR  \
38           MG Bevan (AUS)  232   196  67  6912  108  53.58  9320
74.16
```

```
78  Abdul Razzaq (Asia/PAK)  265   228  57  5080  112  29.70  6252
81.25
68           DR Martyn (AUS)  208   182  51  5346  144  40.80  6877
77.73
53            JN Rhodes (SA)  245   220  51  5935  121  35.11  7336
80.90
64          MEK Hussey (AUS)  185   157  44  5442  109  48.15  6243
87.16

    100  50   0  Exp  Cluster_Id
38    6  46   5   10           3
78    3  23  14   15           3
68    5  37  10   14           3
53    2  33  12   11           3
64    3  39   3    8           3
```

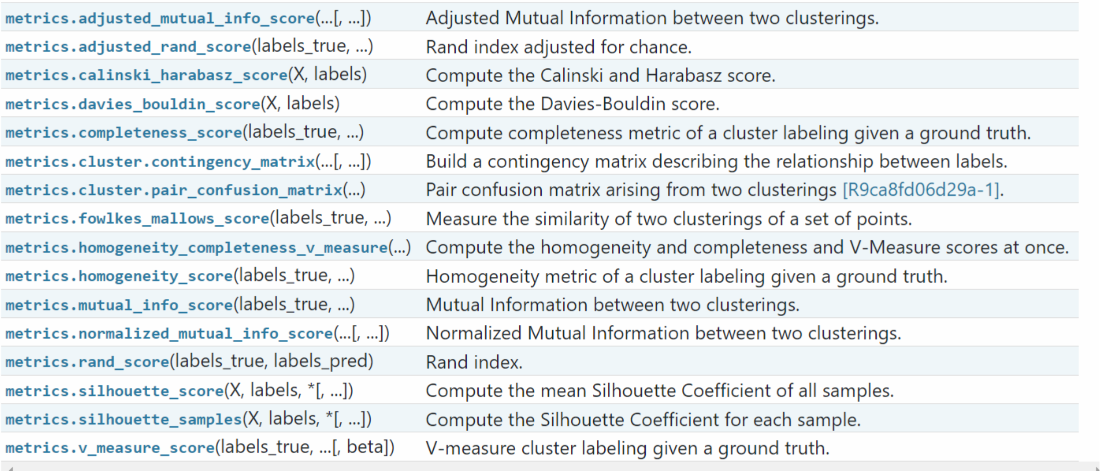
References:

1. Sklearn Clustering : https://scikit-learn.org/stable/modules/clustering.html

2. Kaggle : https://www.kaggle.com/

3. StandardScaler : https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#sklearn.preprocessing.StandardScaler

4. Normalisation : https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.minmax_scale.html#sklearn.preprocessing.minmax_scale



metrics.adjusted_mutual_info_score(...[, ...])	Adjusted Mutual Information between two clusterings.
metrics.adjusted_rand_score(labels_true, ...)	Rand index adjusted for chance.
metrics.calinski_harabasz_score(X, labels)	Compute the Calinski and Harabasz score.
metrics.davies_bouldin_score(X, labels)	Compute the Davies-Bouldin score.
metrics.completeness_score(labels_true, ...)	Compute completeness metric of a cluster labeling given a ground truth.
metrics.cluster.contingency_matrix(...[, ...])	Build a contingency matrix describing the relationship between labels.
metrics.cluster.pair_confusion_matrix(...)	Pair confusion matrix arising from two clusterings [R9ca8fd06d29a-1].
metrics.fowlkes_mallows_score(labels_true, ...)	Measure the similarity of two clusterings of a set of points.
metrics.homogeneity_completeness_v_measure(...)	Compute the homogeneity and completeness and V-Measure scores at once.
metrics.homogeneity_score(labels_true, ...)	Homogeneity metric of a cluster labeling given a ground truth.
metrics.mutual_info_score(labels_true, ...)	Mutual Information between two clusterings.
metrics.normalized_mutual_info_score(...[, ...])	Normalized Mutual Information between two clusterings.
metrics.rand_score(labels_true, labels_pred)	Rand index.
metrics.silhouette_score(X, labels, *[, ...])	Compute the mean Silhouette Coefficient of all samples.
metrics.silhouette_samples(X, labels, *[, ...])	Compute the Silhouette Coefficient for each sample.
metrics.v_measure_score(labels_true, ...[, beta])	V-measure cluster labeling given a ground truth.