```
from google.colab import files
uploaded=files.upload()
```

Choose Files cricket clean.csv

- **cricket clean.csv**(text/csv) - 5470 bytes, last modified: 7/25/2025 - 100% done

```
Saving cricket clean.csv to cricket clean.csv
```

Importing Libraries

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

Loading Dataset

```
df=pd.read_csv('cricket clean.csv')
```

Player 🥷 → The name of the player.

Mat 🏏 → Total number of matches played by the player.

Inns 🎯 → Total number of innings the player has batted in.

NO (Not Outs) 🚫 → Number of times the player remained not out at the end of an innings.

Runs 🏃 → Total runs scored by the player in their career.

HS (Highest Score) 🔝 → The player's highest individual score in a single innings.

Ave (Batting Average) 📊 → The batting average, calculated as total runs divided by number of times out. Ave = Runs / (Inns - NO)

BF (Balls Faced) 🏸 → Total number of balls faced by the player while batting.

SR (Strike Rate) ⚡ → The strike rate, showing how quickly the player scores. SR = (Runs / BF) * 100

100 (Centuries) 💯 → Number of times the player scored 100 or more runs in an innings.

50 (Half-Centuries) 🟡 → Number of times the player scored between 50 and 99 runs in an innings.

0 (Ducks) 🦆 → Number of times the player got out without scoring any runs.

Exp (Experience) ⏳ → The experience level of the player, which can be based on matches played, years active, or any predefined value representing seniority.

EDA

```
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     object 
 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(9), object(2)
memory usage: 8.2+ KB
```

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

```
np.int64(0)
```

```
df
```

	Player	Mat	Inns	NO	Runs	HS	Ave	BF	SR	100	50	0	exp
0	SR Tendulkar (INDIA)	463	452	41	18426	200*	44.83	21367	86.23	49	96	20	23
1	KC Sangakkara (Asia/ICC/SL)	404	380	41	14234	169	41.98	18048	78.86	25	93	15	15
2	RT Ponting (AUS/ICC)	375	365	39	13704	164	42.03	17046	80.39	30	82	20	17
3	ST Jayasuriya (Asia/SL)	445	433	18	13430	189	32.36	14725	91.20	28	68	34	22
4	DPMD Jayawardene (Asia/SL)	448	418	39	12650	144	33.37	16020	78.96	19	77	28	17
...	...	...	...	...	...	...	...	...	...	...	...	...	...
74	CG Greenidge (WI)	128	127	13	5134	133*	45.03	7908	64.92	11	31	3	16
75	Misbah-ul-Haq (PAK)	162	149	31	5122	96*	43.40	6945	73.75	0	42	6	13
76	PD Collingwood (ENG)	197	181	37	5092	120*	35.36	6614	76.98	5	26	7	10
77	A Symonds (AUS)	198	161	33	5088	156	39.75	5504	92.44	6	30	15	11
78	Abdul Razzaq (Asia/PAK)	265	228	57	5080	112	29.70	6252	81.25	3	23	14	15

79 rows × 13 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df['HS']=df['HS'].str.replace("*","")
```

```
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: 8.2+ KB
```

```
df.duplicated().sum()
```

```
np.int64(0)
```

```
cricket=df.copy()
```

```
cricket.drop(['Player'],axis=1,inplace=True)
```

```
cricket
```

	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
...	...	...	...	...	...	...	...	...	...	...	...	...
74	128	127	13	5134	133	45.03	7908	64.92	11	31	3	16
75	162	149	31	5122	96	43.40	6945	73.75	0	42	6	13
76	197	181	37	5092	120	35.36	6614	76.98	5	26	7	10
77	198	161	33	5088	156	39.75	5504	92.44	6	30	15	11
78	265	228	57	5080	112	29.70	6252	81.25	3	23	14	15

79 rows × 12 columns

Next steps: Generate code with cricket | View recommended plots | New interactive sheet

Standardization

```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X = sc.fit_transform(cricket)

X
```

```
         9.59464833e-01,  1.74132756e+00,  6.24019419e-01]
         -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]])
```

```
#converting cricket_sc into DataFraeme
X=pd.DataFrame(X,columns=cricket.columns)
X
```

	Mat	Inns	NO	Runs	HS	Ave	BF	SR	100	50	0	exp
0	2.955282	3.169333	0.764963	4.262328	1.632443	1.072294	3.681214	0.703152	4.656726	3.050057	1.145837	2.812787
1	2.155179	2.138915	0.764963	2.609117	0.635224	0.587725	2.635385	-0.044139	1.671888	2.865418	0.296671	0.380938
2	1.761908	1.924245	0.625397	2.400099	0.474382	0.596226	2.319651	0.110997	2.293729	2.188406	1.145837	0.988900
3	2.711183	2.897417	-0.840046	2.292041	1.278591	-1.047909	1.588295	1.207091	2.044992	1.326755	3.523501	2.508806
4	2.751866	2.682747	0.625397	1.984430	-0.168986	-0.876185	1.996354	-0.034000	0.925678	1.880674	2.504502	0.988900
...	...	...	...	...	...	...	...	...	...	...	...	...
74	-1.587674	-1.481860	-1.188961	-0.979677	-0.522838	1.106299	-0.559768	-1.457604	-0.069268	-0.950465	-1.741328	0.684919
75	-1.126598	-1.167010	0.067133	-0.984410	-1.713068	0.829159	-0.863213	-0.562275	-1.437319	-0.273453	-1.231828	-0.227024
76	-0.651961	-0.709047	0.485831	-0.996241	-0.941027	-0.537836	-0.967512	-0.234765	-0.815478	-1.258197	-1.061995	-1.138967
77	-0.638400	-0.995274	0.206699	-0.997819	0.217035	0.208570	-1.317278	1.332823	-0.691110	-1.012011	0.296671	-0.834986
78	0.270191	-0.036412	1.881490	-1.000973	-1.198374	-1.500174	-1.081580	0.198198	-1.064215	-1.442837	0.126837	0.380938

79 rows × 12 columns

Next steps: Generate code with X View recommended plots New interactive sheet

```
X['50'].mean()
```

```
np.float64(1.658307808933778e-16)
```

```
value  = np.float64(1.658307808933778e-16)
print(f"{value:.20f}")
```

```
0.00000000000000016583
```

```
X['50'].std()
```

```
1.0063898413738648
```

Kmeans model

```
from sklearn.cluster import KMeans

wcss = []

for i in range(1, 8):
  kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
  kmeans.fit(X)  #it will start the clsutering process

  print(kmeans.inertia_)  #printing the wcss values
  wcss.append(kmeans.inertia_)
```

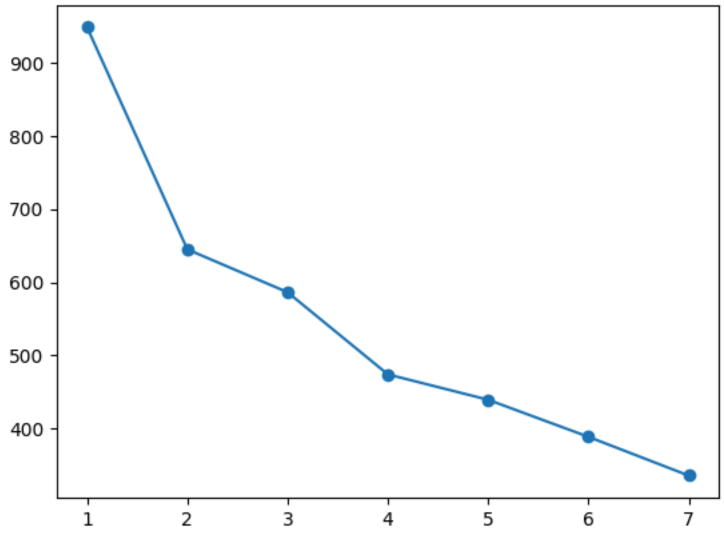
```
948.0000000000002
644.840516217269
```

```
     586.2725236030707
     474.16095045677974
     439.53342325958914
     388.9283315814862
     336.01686079526297
```

```
wcss
```

```
     [948.0000000000002,
      644.840516217269,
      586.2725236030707,
      474.16095045677974,
      439.53342325958914,
      388.9283315814862,
      336.01686079526297]
```

```
plt.plot(range(1,8),wcss,marker='o')
plt.show()
```



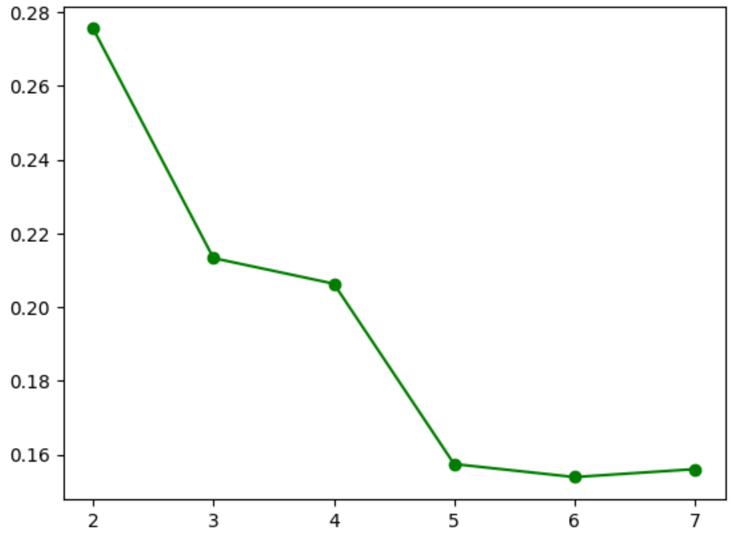
```
from sklearn.metrics import silhouette_score

silhouette_scores=[]

for i in range(2,8):
  kmeans = KMeans(n_clusters =i,random_state=32)
  kmeans.fit(X)

  silhouette_avg = silhouette_score(X,kmeans.labels_)
  silhouette_scores.append(silhouette_avg)
plt.plot(range(2,8),silhouette_scores,marker='o',color = 'green')
```

```
[<matplotlib.lines.Line2D at 0x7e62f05b2050>]
```



```
kmeans = KMeans(n_clusters = 4, init = 'k-means++', random_state = 32)
kmeans.fit(X)  #start the clustering process
```

```
▾            KMeans
KMeans(n_clusters=4, random_state=32)
```

```
y=kmeans.predict(X)
y
```

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

```
df["Clusterid"] = kmeans.labels_
```

```
df
```

	Player	Mat	Inns	NO	Runs	HS	Ave	BF	SR	100	50	0	exp	Clusterid
0	SR Tendulkar (INDIA)	463	452	41	18426	200	44.83	21367	86.23	49	96	20	23	2
1	KC Sangakkara (Asia/ICC/SL)	404	380	41	14234	169	41.98	18048	78.86	25	93	15	15	2
2	RT Ponting (AUS/ICC)	375	365	39	13704	164	42.03	17046	80.39	30	82	20	17	2
3	ST Jayasuriya (Asia/SL)	445	433	18	13430	189	32.36	14725	91.20	28	68	34	22	2
4	DPMD Jayawardene (Asia/SL)	448	418	39	12650	144	33.37	16020	78.96	19	77	28	17	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
74	CG Greenidge (WI)	128	127	13	5134	133	45.03	7908	64.92	11	31	3	16	0
75	Misbah-ul-Haq (PAK)	162	149	31	5122	96	43.40	6945	73.75	0	42	6	13	0
76	PD Collingwood (ENG)	197	181	37	5092	120	35.36	6614	76.98	5	26	7	10	0
77	A Symonds (AUS)	198	161	33	5088	156	39.75	5504	92.44	6	30	15	11	0
78	Abdul Razzaq (Asia/PAK)	265	228	57	5080	112	29.70	6252	81.25	3	23	14	15	0

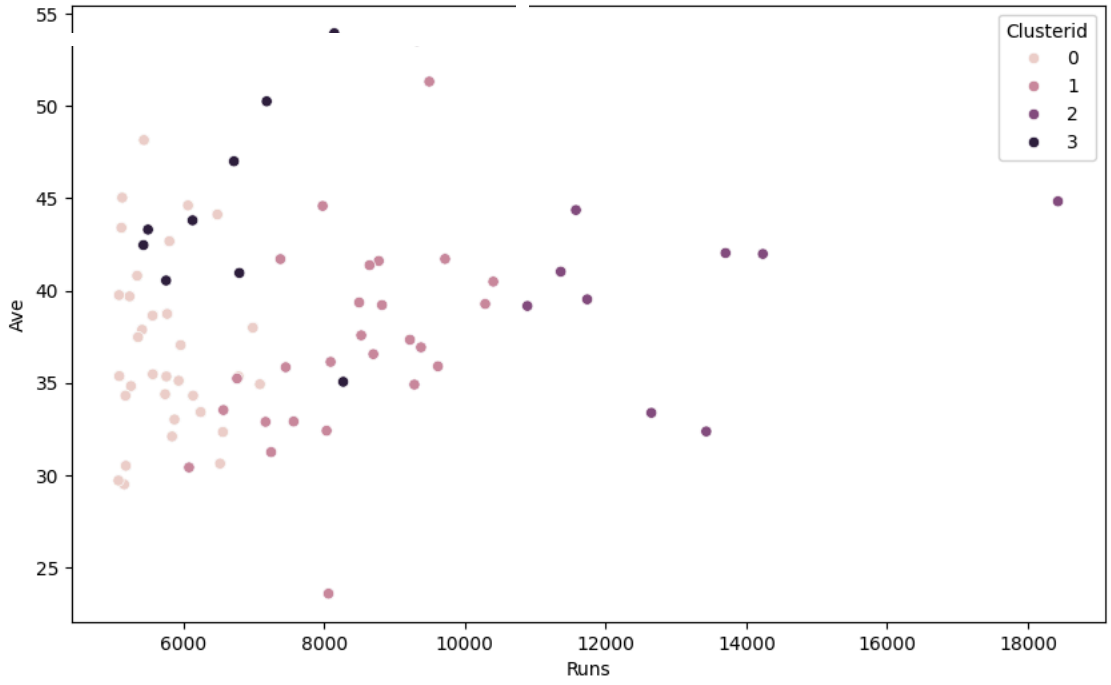
79 rows × 14 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

2d vizualization

```
#2d
plt.figure(figsize=(10,6))
```

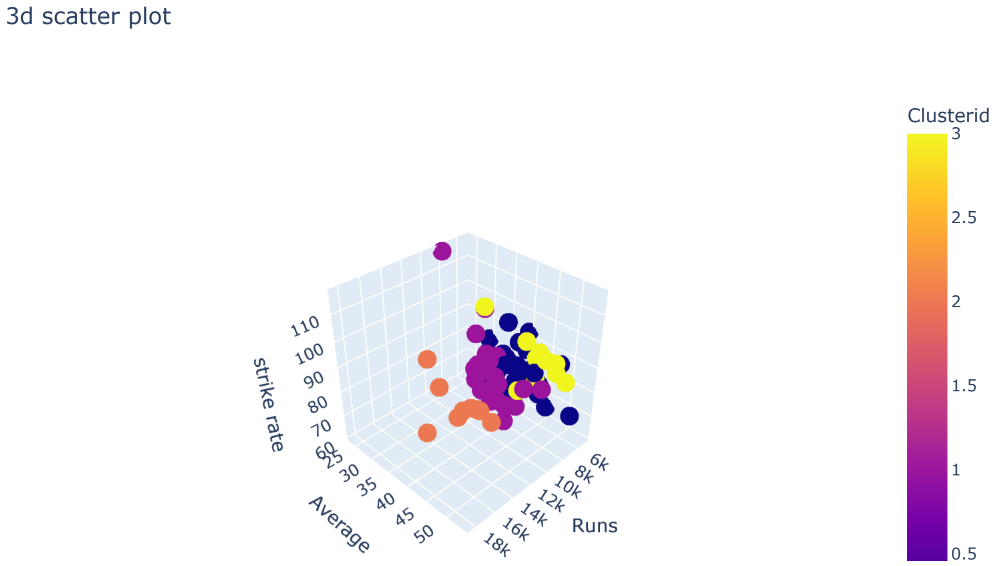
```
sns.scatterplot(data=df,x='Runs',y='Ave',hue='Clusterid'
plt.show()
```



3D vizulaization

```
#3d
import plotly.express as px

fig = px.scatter_3d(df,x="Runs",y="Ave",z='SR',color = "Clusterid",hover_name="Player",title="3d scatter plot")
fig.update_layout(scene = dict(xaxis_title ="Runs",yaxis_title ="Average",zaxis_title='strike rate'),width =800,height = 600)
fig.show()
```


3d scatter plot
strike rate
Average
Runs
6k
8k
10k
12k
14k
16k
18k
Clusterid