Loading the necessary python libraries

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

Loading the dataset

```
df=pd.read_csv(r"/content/VNL2023.csv")
df.head()
```

	Player	Country	Age	Attack	Block	Serve	Set	Dig	Receive	Position
0	Ichikawa Yuki	Japan	28	15.80	1.13	1.40	0.07	4.80	5.60	OH
1	Romano Yuri	Italy	26	12.33	1.07	1.47	0.00	3.87	0.00	OP
2	Abdel-Aziz Nimir	Nederland	31	15.33	0.67	2.08	0.00	3.17	0.25	OP
3	Herrera Jaime Jesus	Cuba	28	15.00	0.92	1.75	0.00	3.33	0.17	OP
4	Takahashi Ran	Japan	22	11.53	0.67	1.00	0.07	6.40	5.07	OH

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

Inspecting the data

```
df.shape
```

```
(131, 10)
```

```
df.describe()
```

	Age	Attack	Block	Serve	Set	Dig	Receive
count	131.000000	131.000000	131.000000	131.000000	131.000000	131.000000	131.000000
mean	27.809160	5.642672	0.845573	0.535802	2.192595	3.428397	1.684198
std	4.186268	4.256229	0.700896	0.454346	6.031587	2.077823	1.989939
min	19.000000	0.000000	0.000000	0.000000	0.000000	0.530000	0.000000
25%	25.000000	2.800000	0.370000	0.240000	0.000000	1.920000	0.000000
50%	27.000000	5.170000	0.690000	0.420000	0.000000	3.000000	0.330000
75%	30.000000	8.600000	1.140000	0.760000	0.000000	4.510000	3.385000
max	41.000000	15.800000	4.080000	2.080000	26.890000	11.440000	6.690000

Describing the dataset in a nutshell and showing the outliers

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

	0
Player	0
Country	0
Age	0
Attack	0
Block	0
Serve	0
Set	0
Dig	0
Receive	0
Position	0

dtype: int64

It shows us that there is no null values in the dataset

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

```
0
```

This shows us that there is no duplicated rows in the dataset

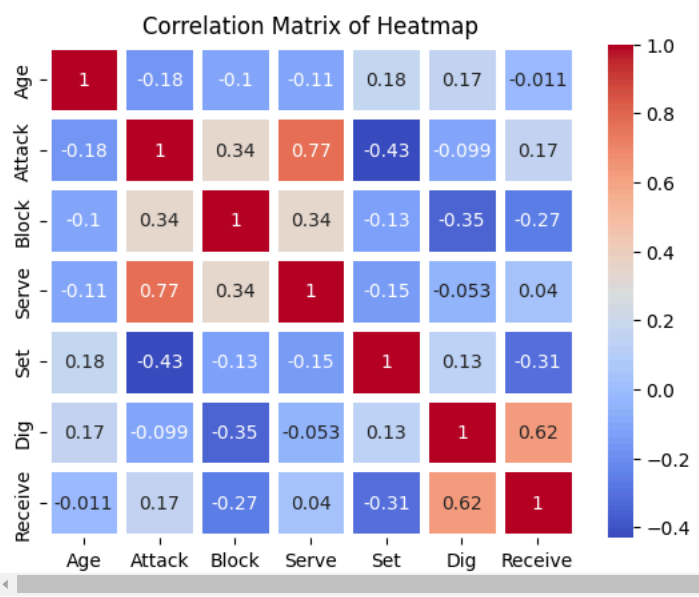
Checking the correlation between colulmns

```
numeric_cols=df.select_dtypes(include={"int","float"}).columns
corr_matrix=df[numeric_cols].corr()
corr_matrix
```

	Age	Attack	Block	Serve	Set	Dig	Receive
Age	1.000000	-0.177849	-0.101040	-0.108367	0.177757	0.167141	-0.011067
Attack	-0.177849	1.000000	0.338412	0.768859	-0.430805	-0.098999	0.169892
Block	-0.101040	0.338412	1.000000	0.335954	-0.132019	-0.348347	-0.265206
Serve	-0.108367	0.768859	0.335954	1.000000	-0.154815	-0.052501	0.039642
Set	0.177757	-0.430805	-0.132019	-0.154815	1.000000	0.131659	-0.305869
Dig	0.167141	-0.098999	-0.348347	-0.052501	0.131659	1.000000	0.624733
Receive	-0.011067	0.169892	-0.265206	0.039642	-0.305869	0.624733	1.000000

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

```
sns.heatmap(corr_matrix,annot=True,cmap="coolwarm",linewidth=5)
plt.title("Correlation Matrix of Heatmap")
plt.show()
```



Above Correlation says :

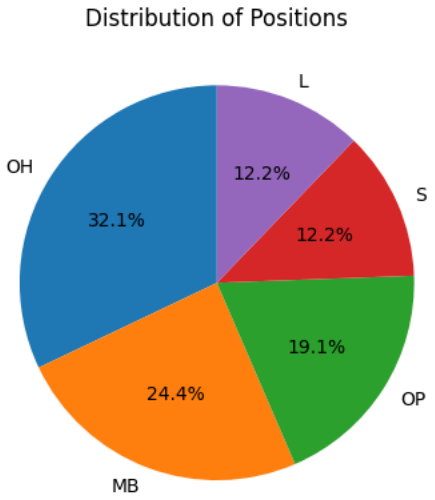
1. Dig and Receive have relations between them
2. Serve and Attack have relation between them

```
position_counts=df["Position"].value_counts()
position_counts
```

Position	count
OH	42
MB	32
OP	25
S	16
L	16

dtype: int64

```
plt.pie(position_counts,labels=position_counts.index,autopct="%1.1f%%",startangle=90)
plt.title("Distribution of Positions")
plt.show()
```



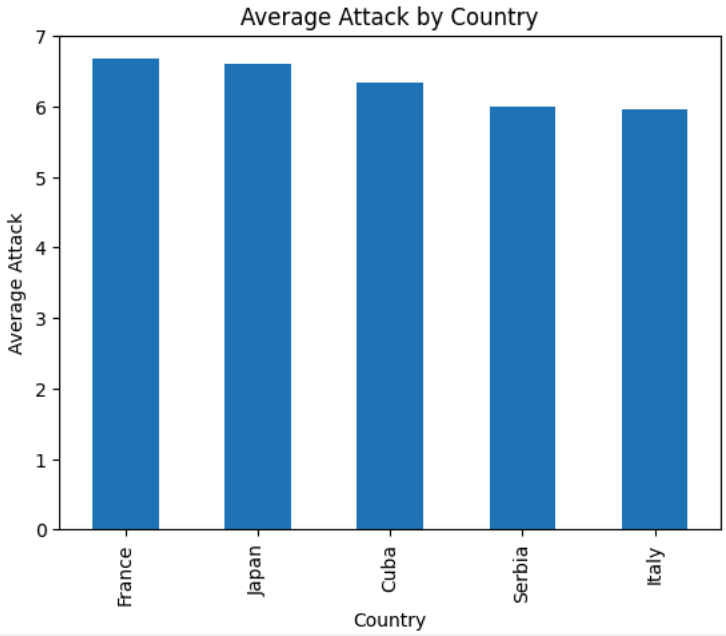

```
avg_attack_by_country=df.groupby("Country")["Attack"].mean()
avg_attack_by_country.sort_values(ascending=False)
```

Country	Attack
France	6.670000
Japan	6.595000
Cuba	6.344286
Serbia	5.998750
Italy	5.965000
Slovenia	5.961250
Argentina	5.925000
Nederland	5.880000
Poland	5.807000
Canada	5.405714
Bulgaria	5.282500
Brazil	5.250000
China	5.093750
Germany	4.833750
Iran	4.707778
USA	4.600000

dtype: float64

```
avg_attack_by_country.sort_values(ascending=False).head(5).plot(kind="bar")
plt.title("Average Attack by Country")
plt.xlabel("Country")
```

```
plt.ylabel("Average Attack")
plt.show()
```




```
avg_serve_by_age=df.groupby("Age")["Serve"].mean()
avg_serve_by_age.sort_values(ascending=False)
```

Age	Serve
31	0.910000
20	0.880000
21	0.770000
26	0.681053
28	0.667273
35	0.666667
27	0.662500
36	0.660000
24	0.640667
22	0.534286
23	0.526667
29	0.477500
30	0.429231
38	0.400000
33	0.321429
32	0.290000
37	0.270000
19	0.200000
25	0.165714
34	0.026667
41	0.000000

dtype: float64

```
df.groupby(["Country","Position"])["Attack"].max().reset_index().sort_values(ascending=False,by="Attack").head(20)
```

	Country	Position	Attack
52	Japan	OH	15.80
58	Nederland	OP	15.33
33	France	OP	15.25
28	Cuba	OP	15.00
68	Serbia	OP	14.33
12	Bulgaria	OH	14.25
22	China	OH	13.50
72	Slovenia	OH	12.62
8	Brazil	OP	12.46
48	Italy	OP	12.33
27	Cuba	OH	12.33
43	Iran	OP	12.00
7	Brazil	OH	11.85
2	Argentina	OH	11.62
62	Poland	OH	10.57
63	Poland	OP	10.57
47	Italy	OH	10.45
38	Germany	OP	10.42
17	Canada	OH	9.75
32	France	OH	9.56

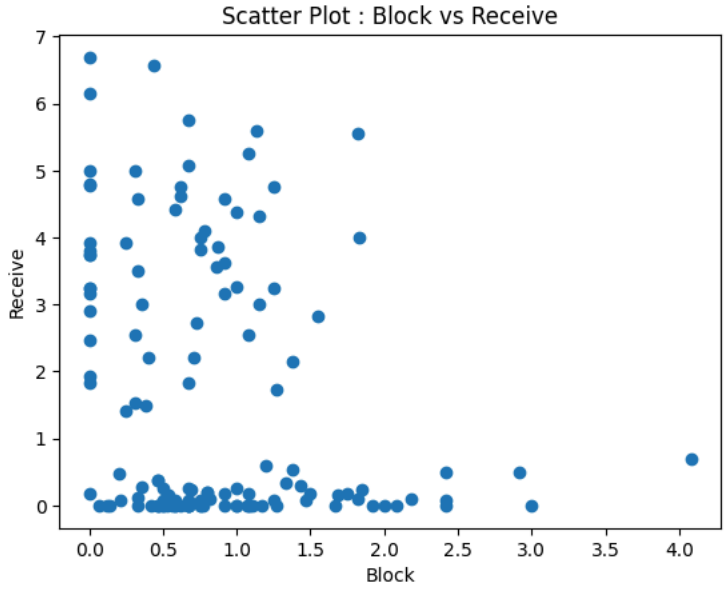
```
df.groupby("Country")["Dig"].sum().sort_values(ascending=False).head(10)
```

Country	Dig
France	38.59
Italy	35.89
Argentina	33.88
Slovenia	33.85
Poland	32.56
Japan	32.38
Serbia	30.64
USA	28.42
Canada	26.50
Brazil	24.61

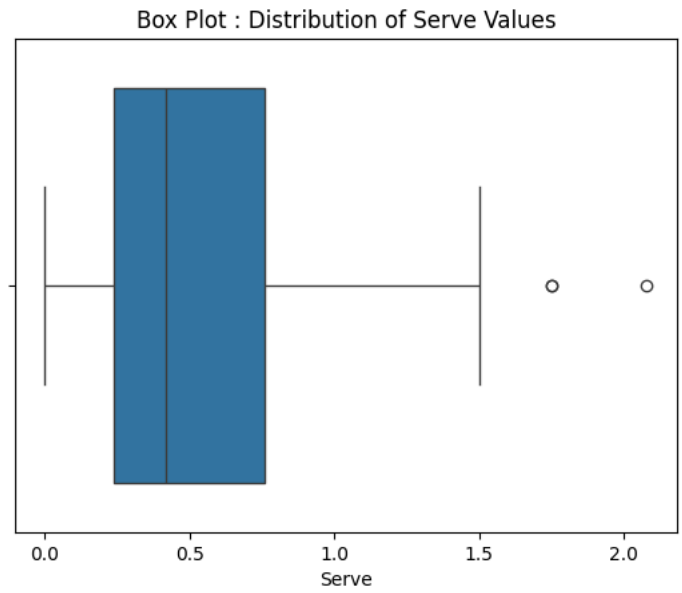
dtype: float64

```
plt.scatter(df["Block"],df["Receive"])
plt.title("Scatter Plot : Block vs Receive")
plt.xlabel("Block")
plt.ylabel("Receive")
```

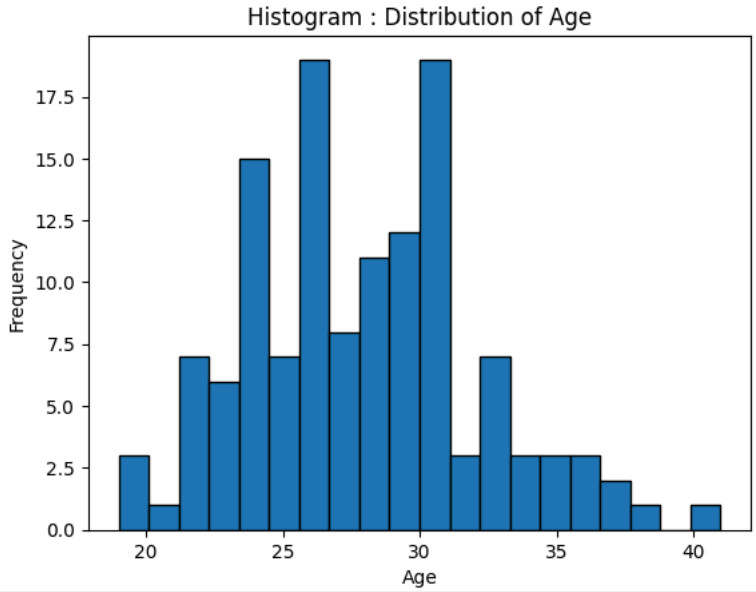
```
Text(0, 0.5, 'Receive')
```



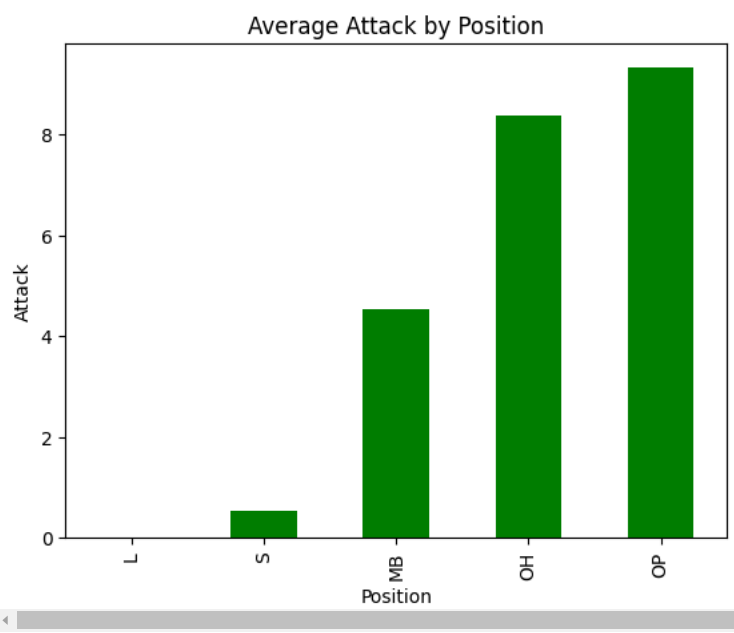

```
sns.boxplot(x=df["Serve"])
plt.title("Box Plot : Distribution of Serve Values")
plt.xlabel("Serve")
plt.show()
```



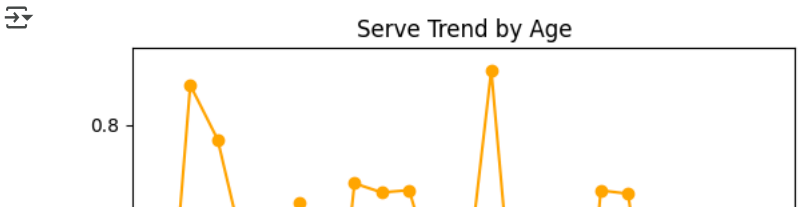

```
plt.hist(df["Age"],bins=20,edgecolor="black")
plt.title("Histogram : Distribution of Age")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.show()
```




```
avg_attack_by_position=df.groupby("Position")["Attack"].mean()
avg_attack_by_position.sort_values(ascending=True).plot(kind="bar",color="green")
plt.title("Average Attack by Position")
plt.xlabel("Position")
plt.ylabel("Attack")
plt.show()
```




```
serve_trend_by_age=df.groupby("Age")["Serve"].mean()
serve_trend_by_age.plot(kind="line",marker="o",linestyle="-",color="orange")
plt.title("Serve Trend by Age")
plt.xlabel("Age")
plt.ylabel("Serve")
plt.show()
```




```
total_attack_block_by_country=df.groupby("Country")[["Attack","Block"]].sum()
total_attack_block_by_country.sort_values(ascending=False,by="Attack").plot(kind="bar",stacked=True,colormap="viridis",figsize=(12,6))
plt.title("Total Attack and Block by Country")
plt.xlabel("Country")
plt.ylabel("Total Attack and Block")
plt.show()
```