Provided summery statistics for a dataset with numeric varibles grouped by one of the qualitative variables.

1. **Mean**: The mean, also known as the average, is a measure of central tendency calculated by summing up all the values in a dataset and dividing the sum by the total number of values.
2. **Median**: The median is another measure of central tendency that represents the middle value of a dataset when it is arranged in ascending or descending order.
3. **Mode**: The mode is the value that appears most frequently in a dataset.

```
import pandas as pd
```

```
df = pd.read_excel('/content/new_data.xlsx')
```

```
df
```

	roll	name	class	marks	age
0	1	Aryan	TE	67.50	23
1	2	Ananya	BE	73.25	21
2	3	Aditya	TE	61.80	25
3	4	Riya	BE	58.90	22
4	5	Vedant	TE	70.10	24
5	6	Aisha	BE	64.70	20
6	7	Arnav	TE	77.60	23
7	8	Sia	BE	69.45	22
8	9	Kabir	TE	66.30	21
9	10	Neha	BE	72.15	24

Next steps: Generate code with `df` View recommended plots

```
#mean
df.mean()
```

```
<ipython-input-4-c61f0c8f89b5>:1: FutureWarning: The default value of numeric_only in DataFrame.mean is deprecated. In a future version, it will de
  df.mean()
roll      5.500
marks    68.175
age      22.500
dtype: float64
```

```
#median
df.median()
```

```
<ipython-input-5-6d467abf240d>:1: FutureWarning: The default value of numeric_only in DataFrame.median is deprecated. In a future version, it will
  df.median()
roll      5.500
marks    68.475
age      22.500
dtype: float64
```

```
#standard deviation
df.std()
```

```
<ipython-input-6-60106f9090b7>:2: FutureWarning: The default value of numeric_only in DataFrame.std is deprecated. In a future version, it will def
  df.std()
roll     3.027650
marks    5.562086
age      1.581139
dtype: float64
```

```
#mode
df.mode()
```

	roll	name	class	marks	age
0	1	Aditya	BE	58.90	21.0
1	2	Aisha	TE	61.80	22.0
2	3	Ananya	NaN	64.70	23.0
3	4	Arnav	NaN	66.30	24.0
4	5	Aryan	NaN	67.50	NaN
5	6	Kabir	NaN	69.45	NaN
6	7	Neha	NaN	70.10	NaN
7	8	Riya	NaN	72.15	NaN
8	9	Sia	NaN	73.25	NaN
9	10	Vedant	NaN	77.60	NaN

```
# max and min function
df.max()
```

```
roll         10
name     Vedant
class        TE
marks      77.6
age          25
dtype: object
```

```
df.min()
```

```
roll          1
name     Aditya
class        BE
marks      58.9
age          20
dtype: object
```

```
import numpy as np
np.std(df['marks'])
```

```
5.276658507047808
```

```
gr1 = df.groupby('class')
```

```
te = gr1.get_group('BE')
# te.min()
te.max()
```

```
roll        10
name       Sia
class       BE
marks    73.25
age         24
dtype: object
```

```
gr2 = df.groupby('age')
gr2.groups
```

```
{20: [5], 21: [1, 8], 22: [3, 7], 23: [0, 6], 24: [4, 9], 25: [2]}
```

```
gr2.get_group(20)
```

	roll	name	class	marks	age
5	6	Aisha	BE	64.7	20

Now using the Seaborn library: Seaborn is a Python data visualization library based on Matplotlib that provides a high-level interface for creating attractive and informative statistical graphics

```
import  seaborn as sns
df = sns.load_dataset('iris')

df
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica

Next Generate code with df View recommended plots

```
gr = df.groupby('species')
se = gr.get_group('setosa')
ve = gr.get_group('versicolor')
vi = gr.get_group('virginica')
```

```
se.shape
```

```
(50, 5)
```

```
ve.shape
```

```
(50, 5)
```

```
vi.shape
```

```
(50, 5)
```

```
se.describe()
```

	sepal_length	sepal_width	petal_length	petal_width
count	50.00000	50.000000	50.000000	50.000000
mean	5.00600	3.428000	1.462000	0.246000
std	0.35249	0.379064	0.173664	0.105386
min	4.30000	2.300000	1.000000	0.100000
25%	4.80000	3.200000	1.400000	0.200000
50%	5.00000	3.400000	1.500000	0.200000
75%	5.20000	3.675000	1.575000	0.300000
max	5.80000	4.400000	1.900000	0.600000

```
vi.describe()
```

	sepal_length	sepal_width	petal_length	petal_width
count	50.00000	50.000000	50.000000	50.00000
mean	6.58800	2.974000	5.552000	2.02600
std	0.63588	0.322497	0.551895	0.27465
min	4.90000	2.200000	4.500000	1.40000
25%	6.22500	2.800000	5.100000	1.80000
50%	6.50000	3.000000	5.550000	2.00000
75%	6.90000	3.175000	5.875000	2.30000
max	7.90000	3.800000	6.900000	2.50000