Practice of statistics of 2nd class lecture

In [ ]:

```
#Combinotarial problrm
# You are coordinating a sales team of 10 people.
#At the end of each quarter, you recognize the Four best-performing salespeople
#How many different podium combinations are possible?
```

In [ ]:

```
#in question  awarding respectively , so order is matter, if it , the cal culati
10!/(10-4)!=5040
```

In [ ]:

```
#If order not matter  follow this
10!/(4!(10-4))!=210
```

#Probanility, is a mathematical branch that deals with uncertainity , the probability of occurance is writeen as #P(A)=1, and not ouccarance , or impossible P(A)=0. so probabilty of events range is 0 to 1 #simple exampleof Flip a coin and the probability of getting head and tail=o.5

In [2]:

```
#To do the probability test we import random module from python
#then we flips the coin with define number with random.choice function
# then calculate the probability of frequncy of head and tail
import random
```

In [3]:

```
flips = random.choices(["H", "T"], k=20)
flips
```

Out[3]:

```
['T',
 'T',
 'T',
 'H',
 'T',
 'T',
 'H',
 'T',
 'T',
 'T',
 'T',
 'H',
 'H',
 'H',
 'T',
 'H',
 'H',
 'H',
 'H',
 'H']
```

In [4]:

```
# from that calculate frequency, total head by total len
fre_H = flips.count("H") / len(flips)
fre_H
```

Out[4]:

```
0.5
```

In [5]:

```
# its actualy a good prediction , its not usual ,if we cahnge the flips no like
flips = random.choices(["H", "T"], k=1000)
flips
```

Out[5]:

```
['H',
 'T',
 'T',
 'H',
 'H',
 'T',
 'T',
 'H',
 'T',
 'T',
 'T',
 'H',
 'T',
 'T',
 'T',
 'H',
 'T',
 'H',
```

In [6]:

```
fre_H = flips.count("H") / len(flips)
fre_H
```

Out[6]:

```
0.487
```

In [7]:

```
#we see its change but close to .50
#Lets check  the tail
fre_T = flips.count("T") / len(flips)
fre_T
```

Out[7]:

```
0.513
```

In [8]:

```
#Lets do the probability with dice
# we know dice have 6 spaces, so define it and folow the filps and count the fre
dice_space = [1, 2, 3, 4, 5, 6]
```

In [9]:

```
rolls = random.choices(dice_space, k=12)
rolls
```

Out[9]:

```
[4, 3, 5, 4, 1, 4, 3, 1, 4, 2, 4, 4]
```

In [11]:

```
fre1_fours = rolls.count(4) / len(rolls)
fre1_fours
```

Out[11]:

```
0.5
```

In [12]:

```
# you see the value is not close to our expected 1/6
#Lets try with large number flip
rolls = random.choices(dice_space, k=1000)
rolls
```

Out[12]:

```
[4,
 6,
 5,
 2,
 3,
 2,
 4,
 3,
 5,
 6,
 2,
 4,
 1,
 2,
 1,
 5,
 5,
 5,
```

In [13]:

```
fre2_fours = rolls.count(4) / len(rolls)
fre2_fours
```

Out[13]:

```
0.168
```

In [ ]:

```
#Insight: The Law of Large Numbers tells us that:

#The relative proportion of an event tends to theoretical probability when we re
```

In [14]:

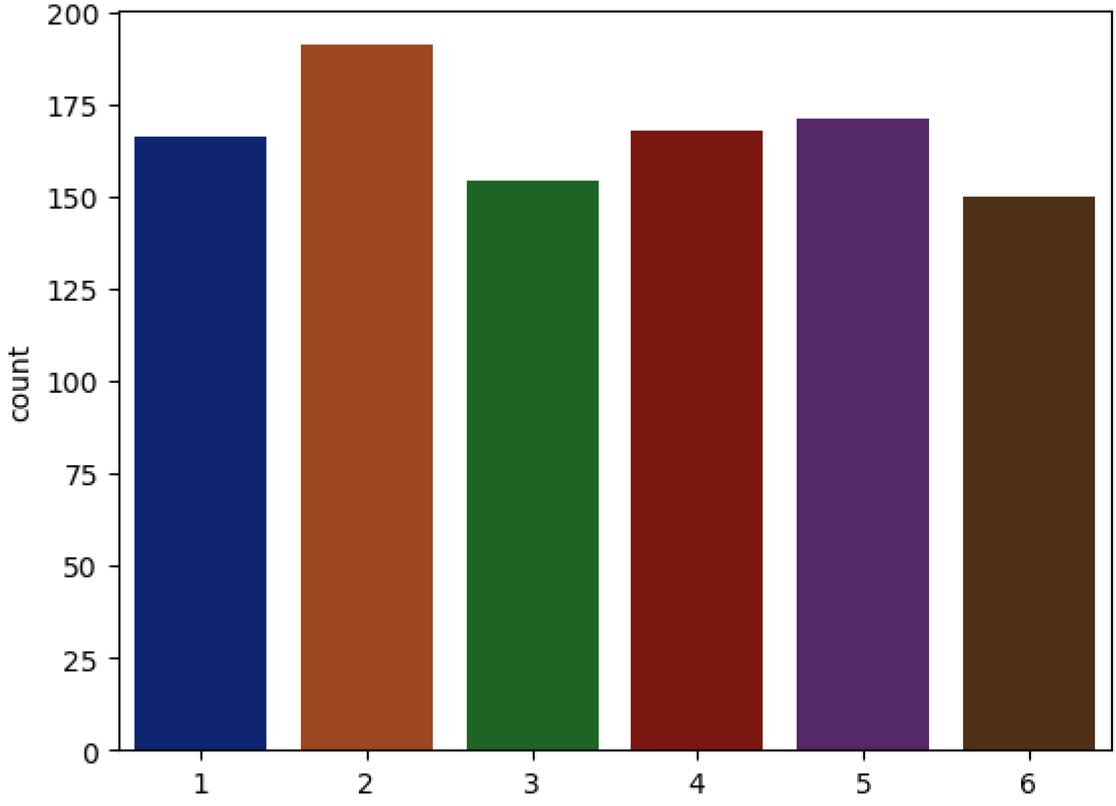
```
#Lets make the plot how its look like. need import seaborn as sns
import seaborn as sns
```

In [18]:

```
sns.countplot(x=rolls, palette="dark")
```

Out[18]:

```
<Axes: ylabel='count'>
```



In [ ]:

```
#insight , in countplot we  see the count or frequency of each dice space , ex 2
```

Set operations

A and B are events

$A \cap B$ (A intersection B) is another event, which occurs if both A and B occur: AND#if both events occurs

$A \cup B$ (A union B) is another event, which occurs if either of A or B occur: OR#if either one occurs

$A'$is the opposite of A

In [22]:

```
#Example of die
A={1,2,3,4,5,6}
B={3,4,5}
```

In [ ]:

```
A∩B={3,4,5}
A∪B={1,2,3,4,6}
```

In [ ]:

```
#morgan law
P(A∪B)=P(A)+P(B)−P(A∩B)
```

In [24]:

```
#Univariate analysis with random sample
#Lets take a sample of 2000 people with their age and salary
# Then check the univariation with different plot and find the corelation betwee
# Import numpy for makeing the random attribute age and salary
import numpy as np
```

In [25]:

```
Age = np.random.normal(100, 10, 2000)#100 is mean of people and 10 is std
```

In [26]:

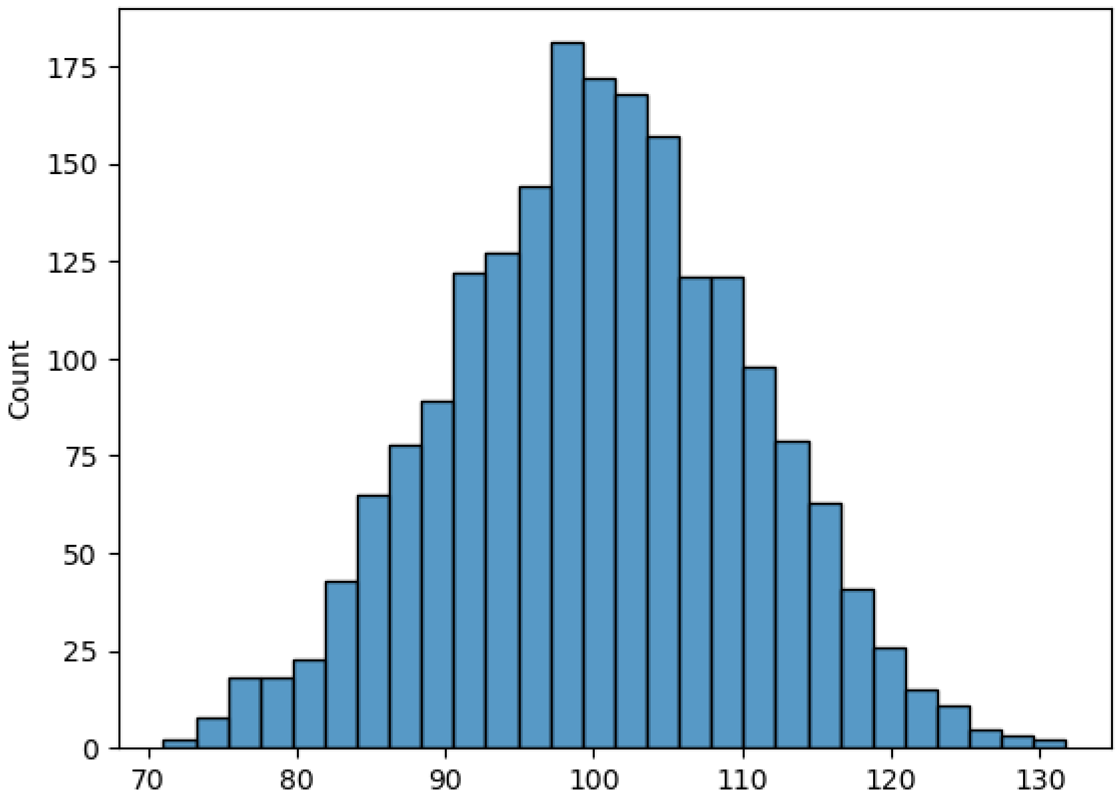
```
salaries = np.random.lognormal(7, 1, 2000)#7 is the mean of the underlying norma
```

In [27]:

```
sns.histplot(x=Age)
```

Out[27]:

```
<Axes: ylabel='Count'>
```



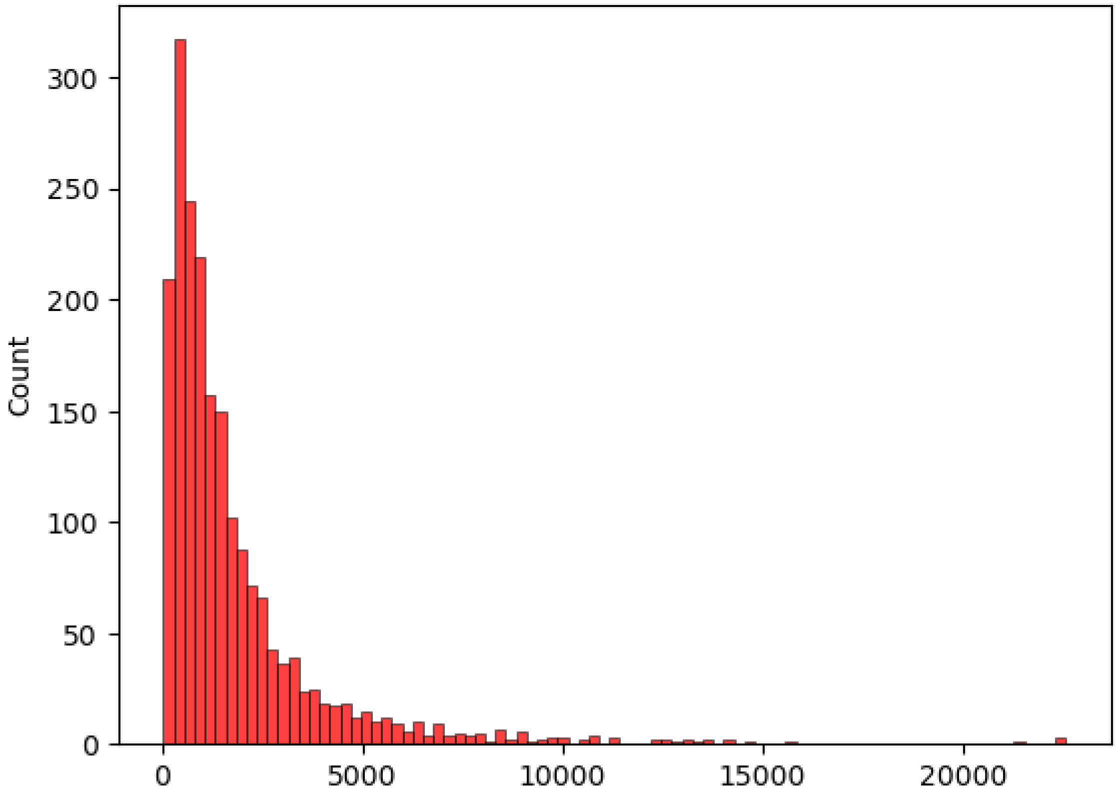
Insight: age attribute showing the normal distribution , that means central tendency is same , and its important variable Letrs try with differnt color with salaries

In [31]:

```
sns.histplot(x=salaries, color="red")
```

Out[31]:

```
<Axes: ylabel='Count'>
```



Insight:its show that the distribution is positiviely skewed , that means theere is outliar, so need to taölk with manager , we can assume mean is greater than median and mode

Arithmatic calculation and visualization

In [32]:

```
Age.mean().round(1)#round the result to one decimal
```

Out[32]:

```
100.3
```

In [33]:

```
salaries.mean().round(1)
```

Out[33]:

```
1806.7
```

In [34]:

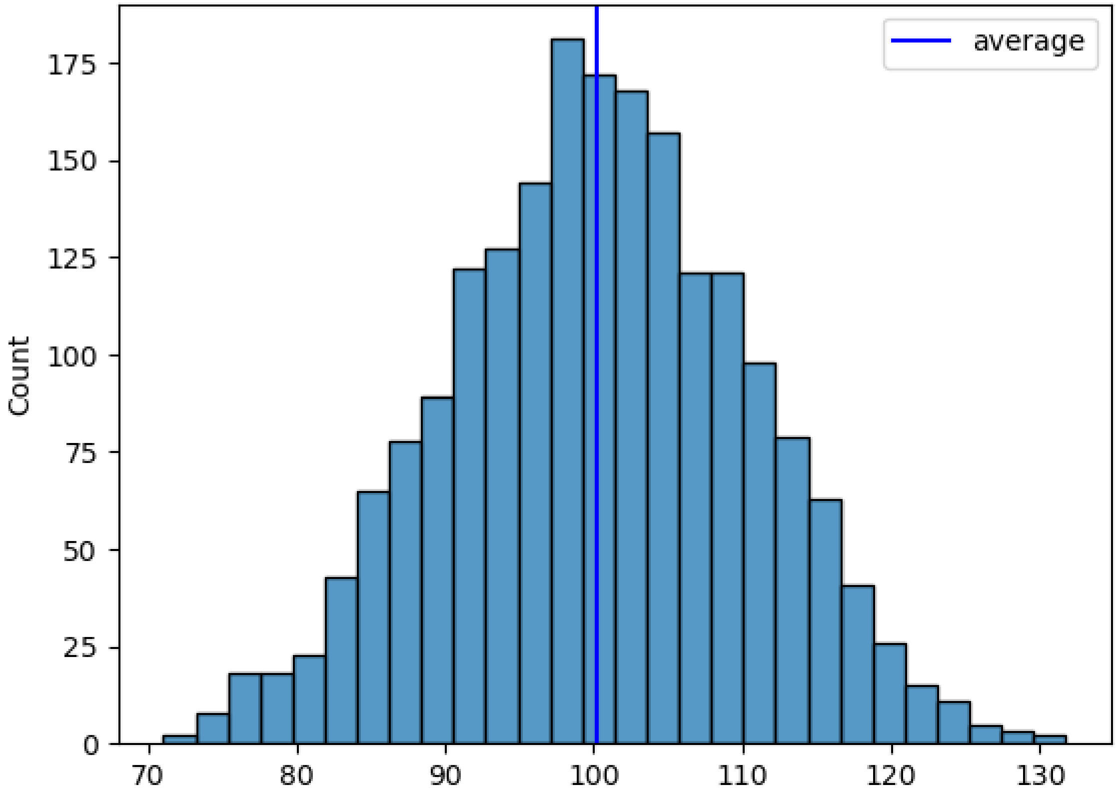
```
#Matplotlib is needed for additional plot customization and features that are nc
# so import matplotlib
import matplotlib.pyplot as plt
```

In [38]:

```
fig, ax = plt.subplots()#define the fig where we can customize by plt.subplot()
g = sns.histplot(x=Age)#then make plot
g.axvline(Age.mean(), c="blue", label="average")#customize rhe plot
plt.legend()
```

Out[38]:

```
<matplotlib.legend.Legend at 0x7fbabc039ff0>
```

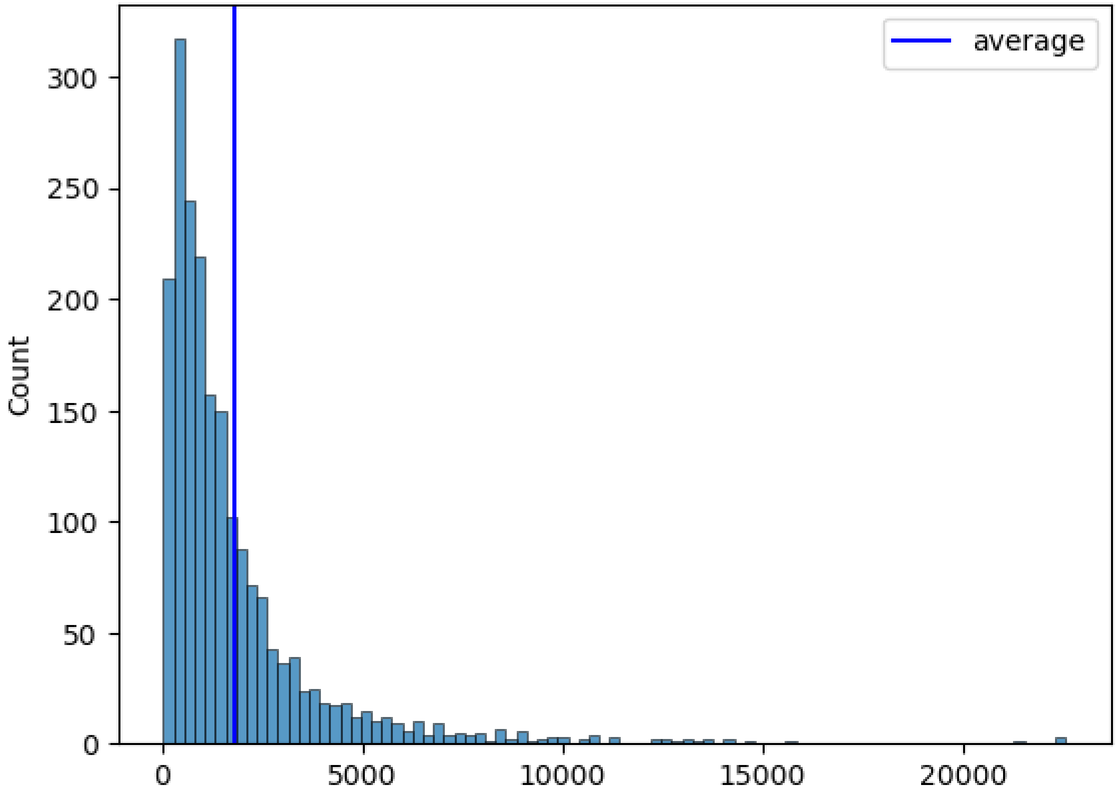


In [39]:

```
fig, ax = plt.subplots()#define the fig where we can customize by plt.subplot()
g = sns.histplot(x=salaries)#then make plot
g.axvline(salaries.mean(), c="blue", label="average")#customize rhe plot in ax
plt.legend()
```

Out[39]:

<matplotlib.legend.Legend at 0x7fbabc5e17e0>



In [40]:

```
#Median
np.median(Age).round(1)
```

Out[40]:

100.3

In [41]:

```
np.median(salaries).round(1)
```

Out[41]:

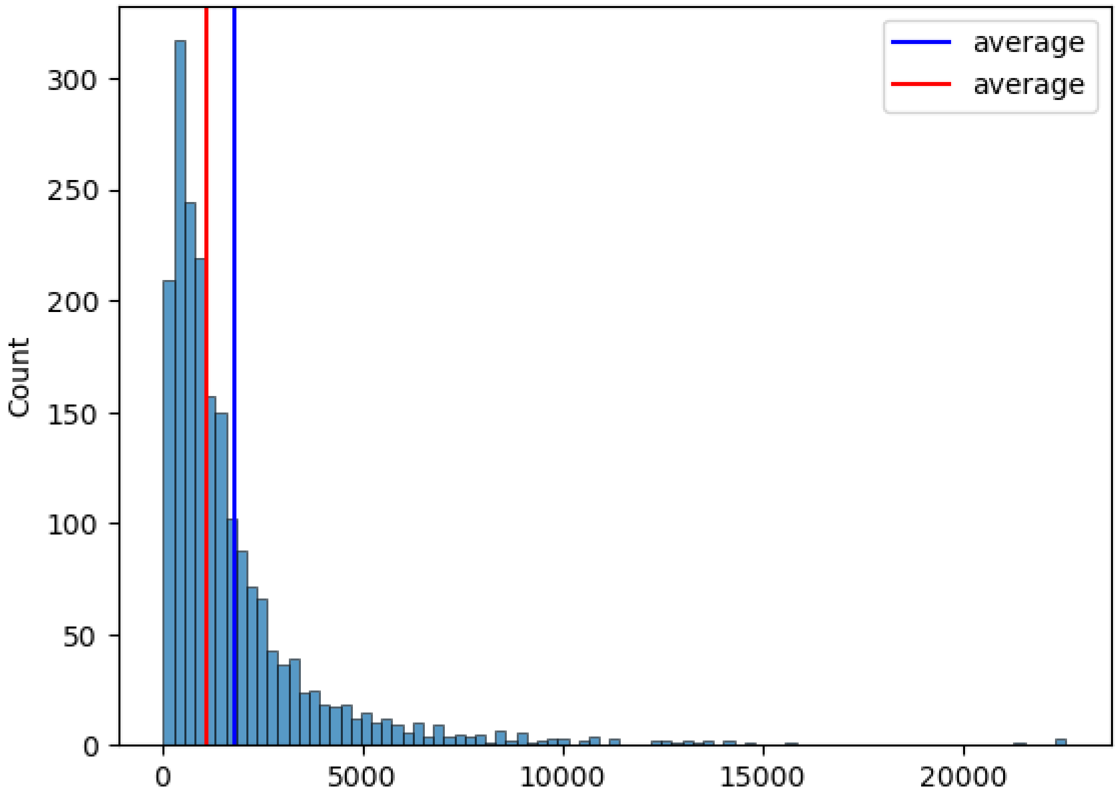
1098.0

In [43]:

```
# we can visualize meaan and median in same plot
fig, ax = plt.subplots()#define the fig where we can customize by plt.subplot()
g = sns.histplot(x=salaries)#then make plot
g.axvline(salaries.mean(), c="blue", label="average")#customize rhe plot
g.axvline(np.median(salaries), c="red", label="average")
plt.legend()
```

Out[43]:

```
<matplotlib.legend.Legend at 0x7fbabc94a1d0>
```

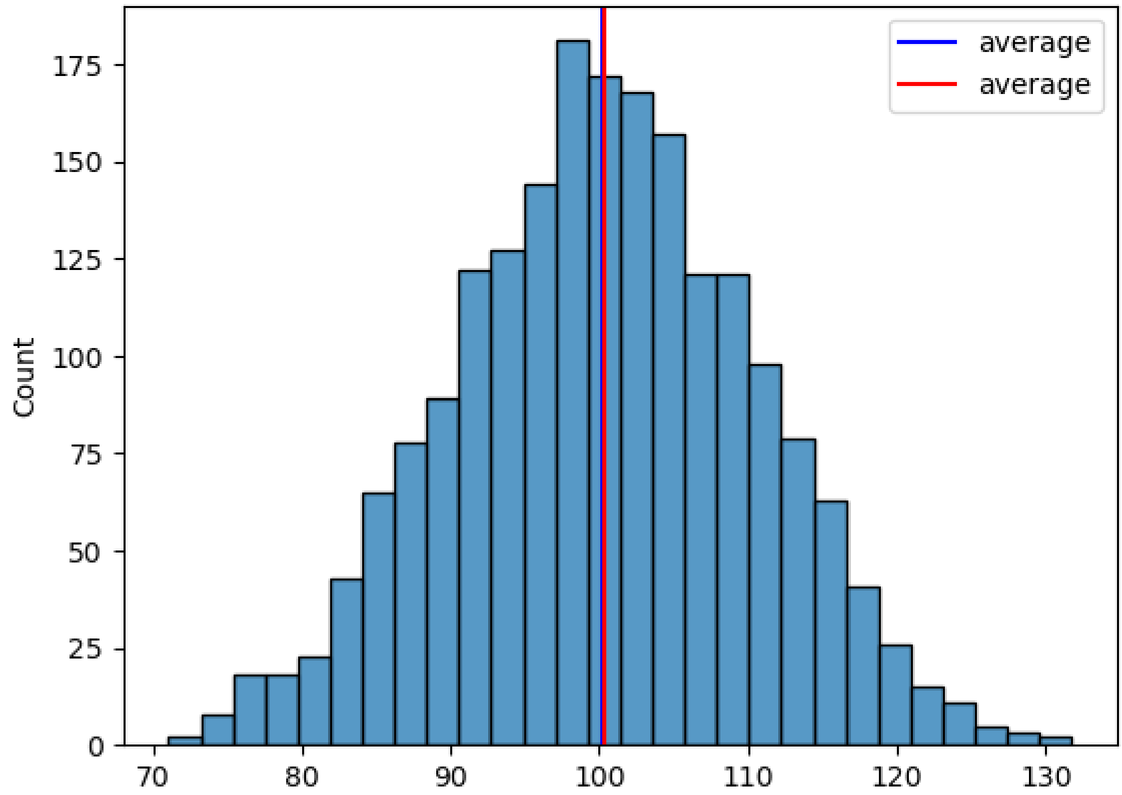


In [44]:

```
# we can visualize meaan and median in same plot
fig, ax = plt.subplots()#define the fig where we can customize by plt.subplot()
g = sns.histplot(x=Age)#then make plot
g.axvline(Age.mean(), c="blue", label="average")#customize rhe plot
g.axvline(np.median(Age), c="red", label="average")
plt.legend()
```

Out[44]:

```
<matplotlib.legend.Legend at 0x7fbabc9d3ee0>
```



Insight:we see in age attribute mean median same , no outliar, where salary mean is greater than median and have outliar

In [ ]:

```
# Variance and std.dev- that helps to know element of attribute , is they are ir
# Lets take two variavle which contain 500 random point  ehich std.dev different

```

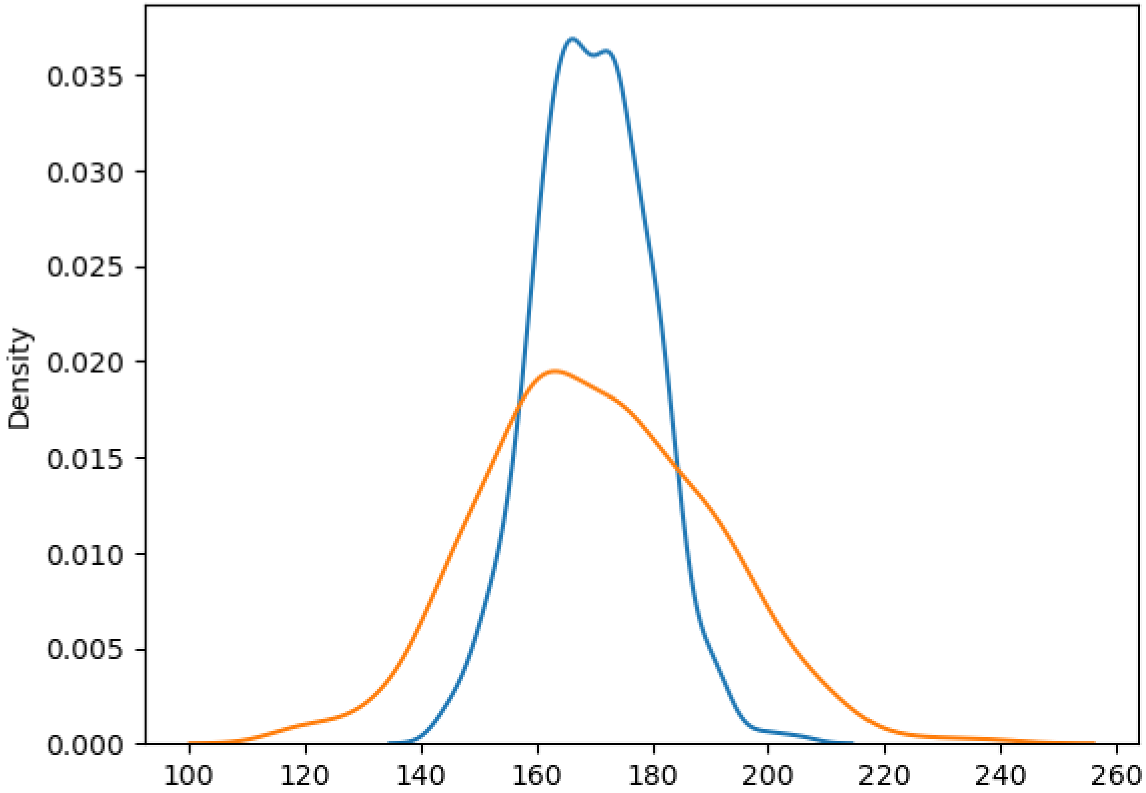
In [45]:

```
N = 500

sns.kdeplot(x=np.random.normal(170, 10, N))
sns.kdeplot(x=np.random.normal(170, 20, N))
```

Out[45]:

<Axes: ylabel='Density'>



In [ ]:

```
#insight: you see how std.dev have effects in distribution
```

In [46]:

```
#A percentile indicates the value below which a certain percentage of observatio

#For example, the 50th percentile is the value (or score) below where 50% of the
np.quantile(salaries, 0.5).round(1)
```

Out[46]:

1098.0

In [47]:

```
np.quantile(salaries, 0.1).round(1)
```

Out[47]:

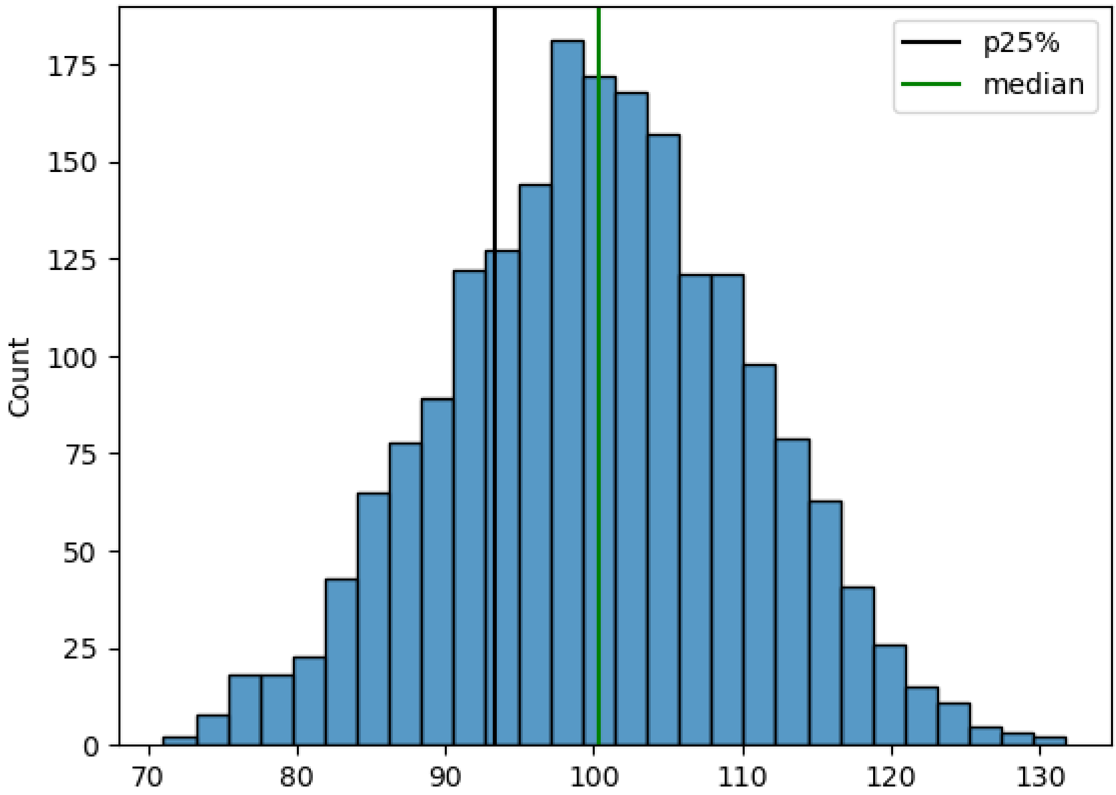
278.3

In [50]:

```
fig, ax = plt.subplots()
g = sns.histplot(x=Age)
g.axvline(np.quantile(Age, 0.25), c="black", label="p25%")
g.axvline(np.median(Age), c="green", label="median")
plt.legend()
```

Out[50]:

```
<matplotlib.legend.Legend at 0x7fbabd02ecb0>
```



insight:25% of data fall in the left side of distribution

In [51]:

```
#Lets convert the attribute of array into series and calculate the quantile in r
import pandas as pd
pd.Series(salaries).quantile(np.linspace(0, 1, 10)).round(1)
```

Out[51]:

```
0.000000       28.9
0.111111      300.9
0.222222      482.2
0.333333      691.5
0.444444      936.2
0.555556     1281.9
0.666667     1676.6
0.777778     2357.9
0.888889     3763.0
1.000000    22566.9
dtype: float64
```

insight: For example, the quantile at 10% indicates that 10% of the salaries are below that value

In [ ]:

```
# Corelation between two varaibles
```

In [53]:

```
df=pd.read_csv("/Users/myyntiimac/Desktop/EMP SAL.csv")
df.head()
```

Out[53]:

	Position	Level	Salary
0	Business Analyst	1	45000
1	Junior Consultant	2	50000
2	Senior Consultant	3	60000
3	Manager	4	80000
4	Country Manager	5	110000

```
df1 = df.drop('position', axis=1)
df1.head()
```

In [55]:

```
df1 = df.drop('Position', axis=1)
df1.head()
```

Out[55]:

	Level	Salary
0	1	45000
1	2	50000
2	3	60000
3	4	80000
4	5	110000

In [56]:

```
df.shape
```

Out[56]:

```
(10, 3)
```

In [57]:

```
# find the corelation and visulize the corelation
df1.corr().round(1)
```

Out[57]:

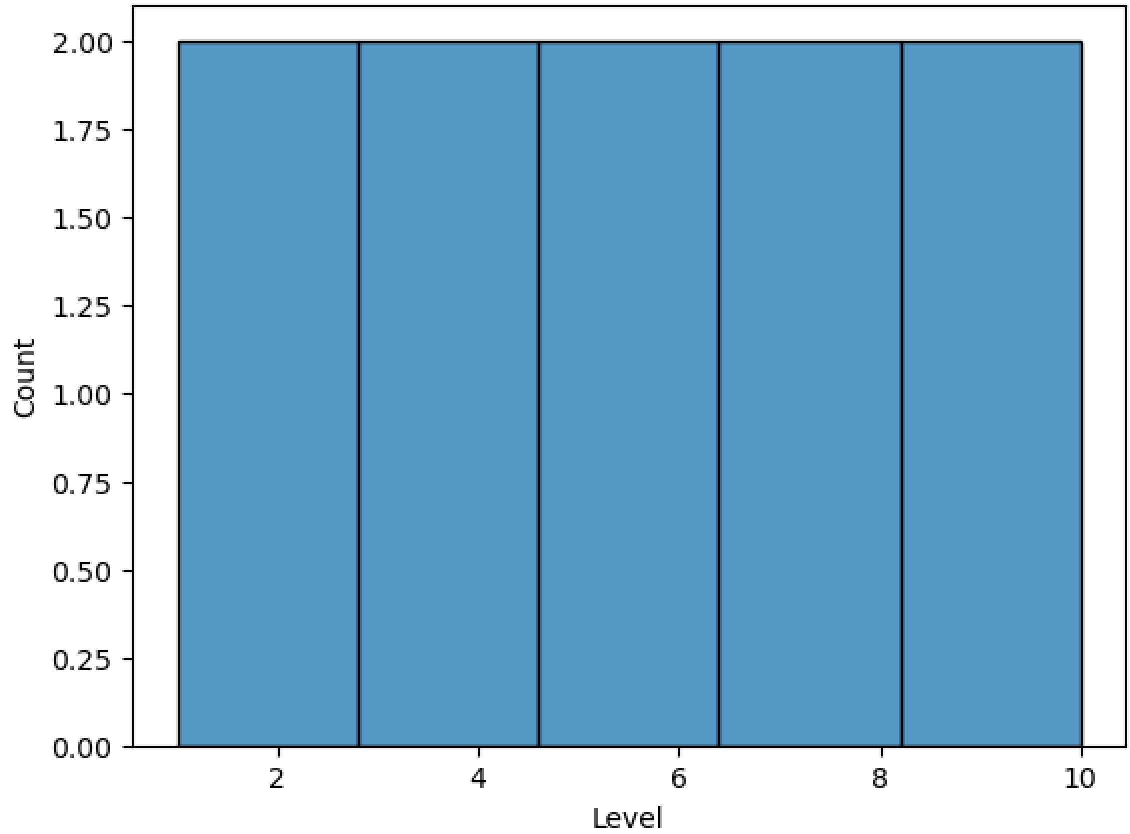
	Level	Salary
Level	1.0	0.8
Salary	0.8	1.0

In [58]:

```
sns.histplot(x=df1.Level)
```

Out[58]:

<Axes: xlabel='Level', ylabel='Count'>

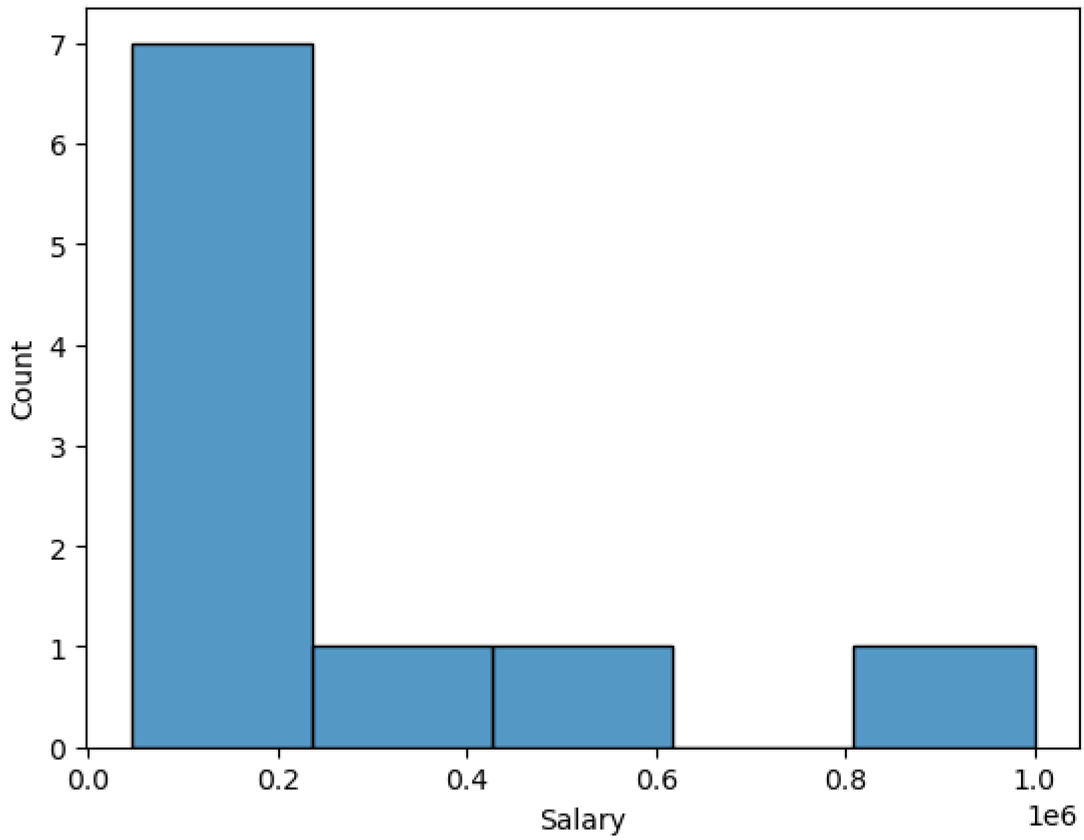


In [59]:

```
sns.histplot(x=df1.Salary)
```

Out[59]:

```
<Axes: xlabel='Salary', ylabel='Count'>
```

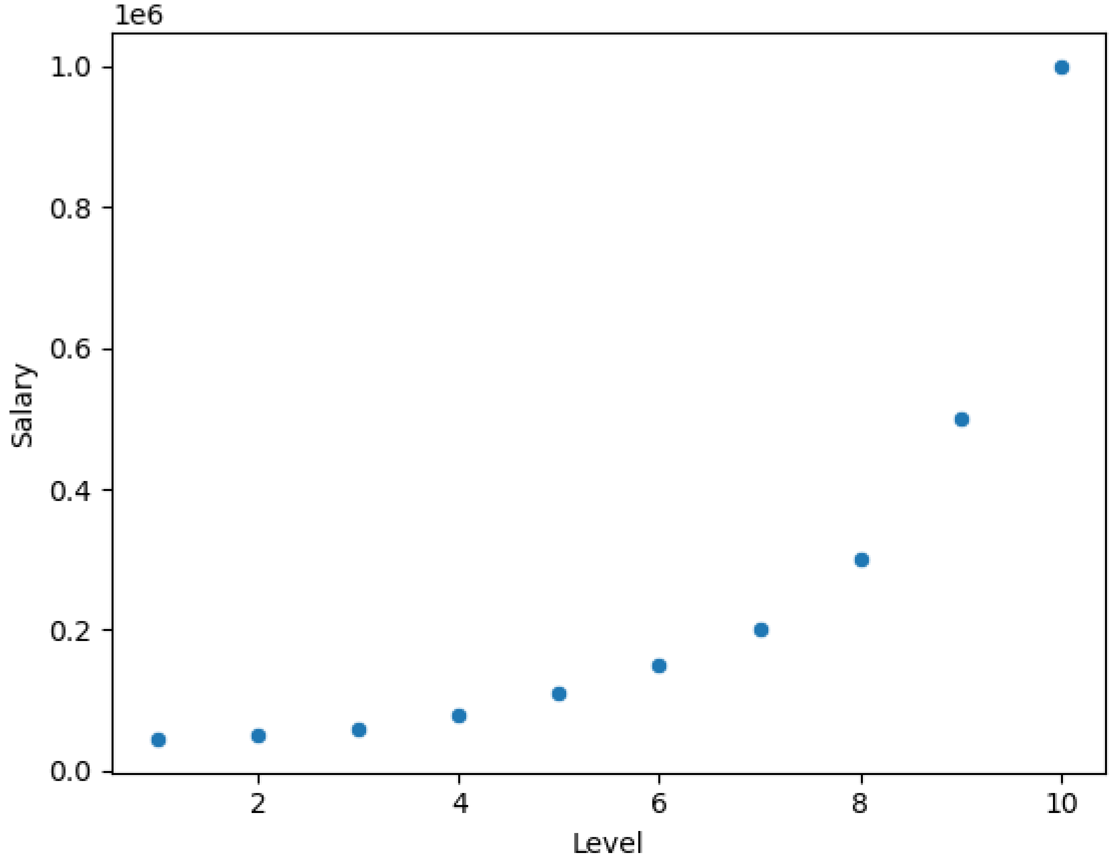


In [60]:

```
sns.scatterplot(x=df1.Level, y=df1.Salary)
```

Out[60]:

```
<Axes: xlabel='Level', ylabel='Salary'>
```



Insight:this scatter plot tells us