1_Measure of Central Tendency

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Mean, Media, Mode are often used in data cleaning

In [3]:
```
dataset = pd.read_csv("titanic.csv")
```

In [4]:
```
dataset.head(3)
```

Out[4]:

	Survived	Pclass	Name	Sex	Age	Siblings/Spouses Aboard	Parents/Children Aboard	Fare
0	0	3	Mr. Owen Harris Braund	male	22.0	1	0	7.2500
1	1	1	Mrs. John Bradley (Florence Briggs Thayer) Cum...	female	38.0	1	0	71.2833
2	1	3	Miss. Laina Heikkinen	female	26.0	0	0	7.9250

In [42]:
```
dataset["Age"]
```

Out[42]:
```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
882    27.0
883    19.0
884     7.0
885    26.0
886    32.0
Name: Age, Length: 887, dtype: float64
```

Find Median

To remove null values in age column

In [5]:
```
# In order to see how many null entries are present in all columns
dataset.isnull().sum()
```

Out[5]:
```
Survived                  0
Pclass                    0
Name                      0
Sex                       0
Age                       0
Siblings/Spouses Aboard   0
Parents/Children Aboard   0
Fare                      0
dtype: int64
```

In [6]:
```
# There are no null values above, in case there are null values we can remove them
dataset["Age"].fillna(dataset["Age"].mean(), inplace=True)
```

In [7]:
```
np.median(dataset["Age"])
```

Out[7]:
```
28.0
```

Find Mean

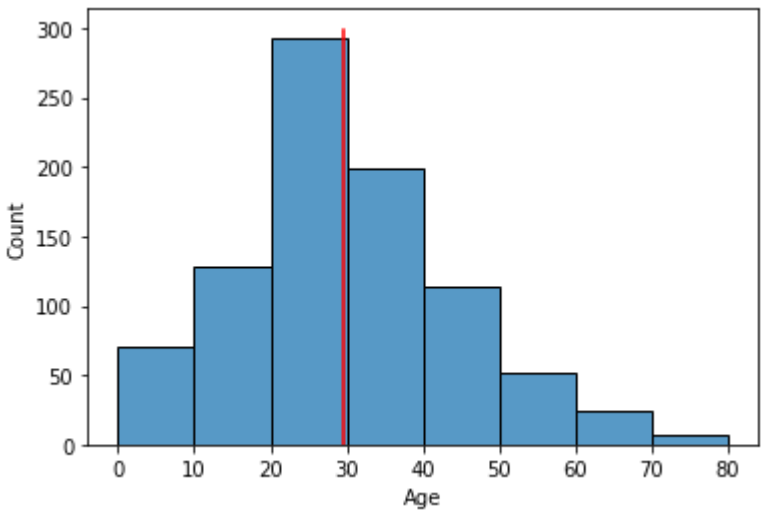
In [8]:
```
dataset["Age"].mean()
```

Out[8]:
```
29.471443066516347
```

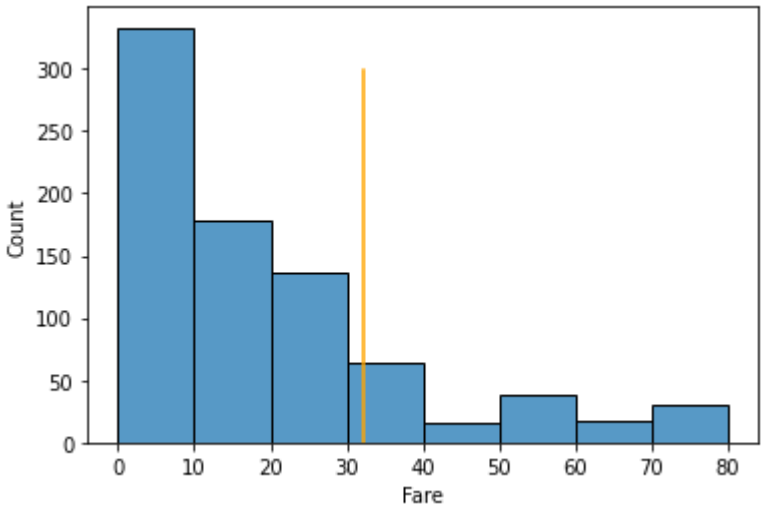
In [13]:
```
mn = np.mean(dataset["Age"])
md = np.mean(dataset["Fare"])
md
```

Out[13]:
```
32.30542018038331
```

In [10]:
```
sns.histplot(x="Age", data=dataset, bins= [i for i in range(0,81,10)])
plt.plot([mn for i in range(0,300)],[i for i in range(0,300)], c="red")
plt.show()
```



In [49]:
```
sns.histplot(x="Fare", data=dataset, bins =[i for i in range(0,81,10)])
plt.plot([md for i in range(0,300)],[i for i in range(0,300)], c="orange")
plt.show()
```



Finding Mode

In [21]:
```
dataset["Fare"].mode()
```

Out[21]:
```
0    8.05
Name: Fare, dtype: float64
```

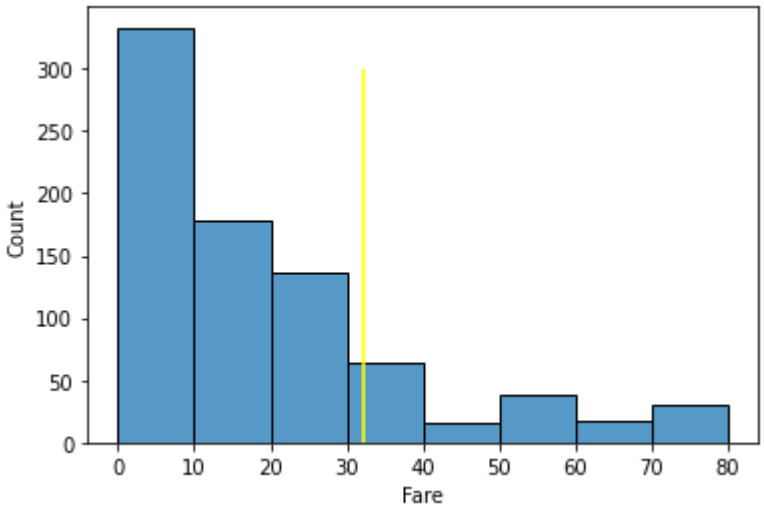
In [27]:
```
mo = dataset["Fare"].mode()[0]
mo
```

Out[27]:
```
8.05
```

In [28]:
```
# To determine the frequency of fare
dataset["Fare"].value_counts()
```

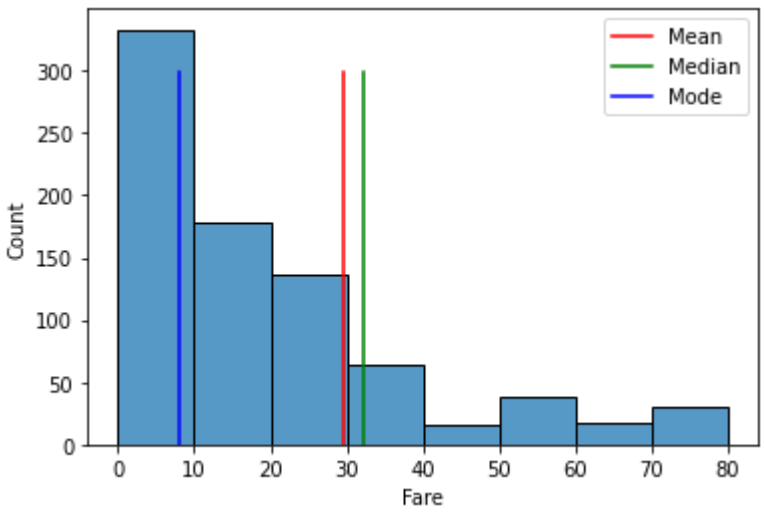
Out[28]:
```
8.0500     43
13.0000    42
7.8958     36
7.7500     33
26.0000    31
           ..
35.0000     1
28.5000     1
6.2375      1
14.0000     1
10.5167     1
Name: Fare, Length: 248, dtype: int64
```

In [48]:
```
# to plot the mode of Fare ind dataset
sns.histplot(x="Fare", data=dataset, bins=[i for i in range(0,81,10)])
plt.plot([md for i in range(0,300)], [i for i in range(0,300)], c="yellow")
plt.show()
```



To show all variables in one plot

In [56]:
```
sns.histplot(x="Fare", data=dataset, bins=[i for i in range(0,81,10)])
plt.plot([mn for i in range(0,300)], [i for i in range(0,300)], c="red", label="Mea
plt.plot([md for i in range(0,300)], [i for i in range(0,300)], c="green", label="M
plt.plot([mo for i in range(0,300)], [i for i in range(0,300)], c="blue", label="Mo
plt.legend()
plt.show()
```



2. Measure of Variability

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
dataset = pd.read_csv('titanic.csv')
```

In [4]:
```
dataset.head(3)
```

Out[4]:

	Survived	Pclass	Name	Sex	Age	Siblings/Spouses Aboard	Parents/Children Aboard	Fare
0	0	3	Mr. Owen Harris Braund	male	22.0	1	0	7.2500
1	1	1	Mrs. John Bradley (Florence Briggs Thayer) Cum...	female	38.0	1	0	71.2833
2	1	3	Miss. Laina Heikkinen	female	26.0	0	0	7.9250

2.1 Range

In [8]:
```
min_r = dataset['Age'].min()
max_r = dataset['Age'].max()
```

In [9]:
```
min_r, max_r
```

Out[9]:
```
(0.42, 80.0)
```

In [10]:
```
range = max_r - min_r
```

In [11]:
```
range
```

Out[11]:
```
79.58
```

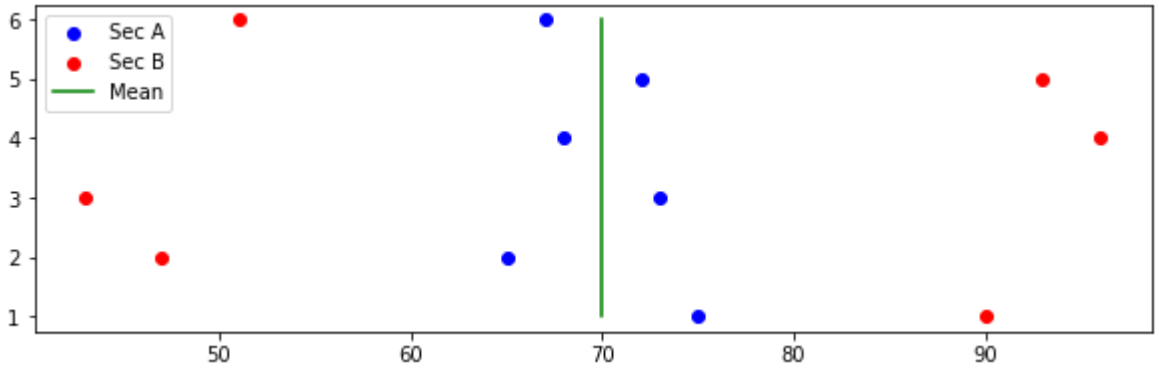
2.2 Mean Absolute Division

To simply print graph

In [23]:
```
sec_a = np.array([75,65,73,68,72,67])
sec_b = np.array([90,47,43,96,93,51])
ne = np.array([1,2,3,4,5,6])
```

In [36]:
```
mean = np.mean(sec_a)
```

In [42]:
```
plt.figure(figsize=(10,3))
plt.scatter(sec_a, ne, color="blue", label="Sec A")
plt.scatter(sec_b, ne, color="red", label="Sec B")
plt.plot([70,70,70,70,70,70], ne, c="green", label="Mean")
#plt.plot([mean for i in range(1,7)], ne, c="green", label="Mean")
plt.legend()
plt.show()
```



To use MAD formula

In [44]:
```
# To calculate xi-x
sec_b - mean
```

Out[44]:
```
array([ 5., -5.,  3., -2.,  2., -3.])
```

In [48]:
```
# To calculate |xi-x|
np.abs(sec_a -mean)
```

Out[48]:
```
array([5., 5., 3., 2., 2., 3.])
```

In [49]:
```
# To calculate sigma|xi-x|
np.sum(np.abs(sec_a - mean))
```

Out[49]:
```
20.0
```

In [51]:
```
# To calculate sigma|xi-x|/n
mad_sec_a = np.sum(np.abs(sec_a - mean))/len(sec_a)
```

In [52]:
```
# Likewise we will calculat mean absolute division of sec_b
mad_sec_b = np.sum(np.abs(sec_b - mean))/len(sec_b)
```

In [53]:
```
mad_sec_a, mad_sec_b
```

Out[53]:
```
(3.3333333333333335, 23.0)
```

So you will take sec_a for machine learning model as it contains low mean absolute division

2.3 Calculate Standard Deviation and Variance

In [55]:
```
# To calculate standard deviation of data of section A and section B
np.std(sec_a), np.std(sec_b)
```

Out[55]:
```
(3.559026084010437, 23.18045153428495)
```

In [56]:
```
# To calculate variance of data of section A and section B
np.var(sec_a), np.var(sec_b)
```

Out[56]:
```
(12.666666666666666, 537.3333333333334)
```

So We will take data of section A becuase it has low variance as well as less standard deviation

To calculate std and var on real world data

In [58]:
```
dataset = pd.read_csv('titanic.csv')
```

In [60]:
```
dataset.head(3)
```

Out[60]:

	Survived	Pclass	Name	Sex	Age	Siblings/Spouses Aboard	Parents/Children Aboard	Fare
0	0	3	Mr. Owen Harris Braund	male	22.0	1	0	7.2500
1	1	1	Mrs. John Bradley (Florence Briggs Thayer) Cum...	female	38.0	1	0	71.2833
2	1	3	Miss. Laina Heikkinen	female	26.0	0	0	7.9250

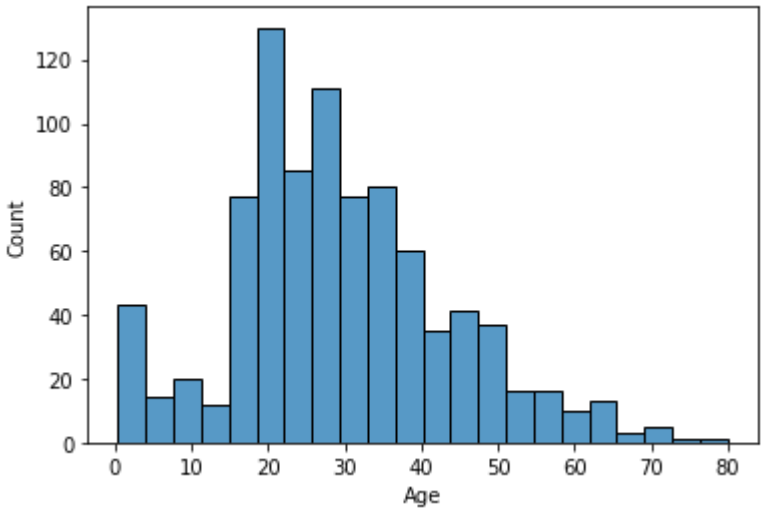
In [61]:
```
dataset['Age'].var()
```

Out[61]:
```
199.42829701227413
```

In [64]:
```
dataset['Age'].std()
```

Out[64]:
```
14.12190840546256
```

In [63]:
```
sns.histplot(x='Age', data=dataset)
plt.show()
```



In [65]:
```
dataset.describe()
```

Out[65]:

	Survived	Pclass	Age	Siblings/Spouses Aboard	Parents/Children Aboard	Fare
count	887.000000	887.000000	887.000000	887.000000	887.000000	887.00000
mean	0.385569	2.305524	29.471443	0.525366	0.383315	32.30542
std	0.487004	0.836662	14.121908	1.104669	0.807466	49.78204
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.00000
25%	0.000000	2.000000	20.250000	0.000000	0.000000	7.92500
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.45420
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.13750
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.32920

In [ ]:
```

```

3_Percentage, Percentile and Quartile

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
dataset = pd.read_csv('titanic.CSV')
```

In [3]:
```
dataset.head(3)
```

Out[3]:

	Survived	Pclass	Name	Sex	Age	Siblings/Spouses Aboard	Parents/Children Aboard	Fare
0	0	3	Mr. Owen Harris Braund	male	22.0	1	0	7.2500
1	1	1	Mrs. John Bradley (Florence Briggs Thayer) Cum...	female	38.0	1	0	71.2833
2	1	3	Miss. Laina Heikkinen	female	26.0	0	0	7.9250

In [5]:
```
dataset.isnull().sum()
```

Out[5]:
```
Survived                  0
Pclass                    0
Name                      0
Sex                       0
Age                       0
Siblings/Spouses Aboard   0
Parents/Children Aboard   0
Fare                      0
dtype: int64
```

In [ ]:
```
# So no null value is present in above data
```

In [7]:
```
np.percentile(dataset['Age'], 25), np.percentile(dataset['Age'], 75)
```

Out[7]:
```
(20.25, 38.0)
```

In [13]:
```
np.percentile(dataset['Age'], 0), np.percentile(dataset['Age'], 100), np.percentile
```

Out[13]:
```
(0.42, 80.0, 28.0)
```

In [14]:
```
dataset['Age'].min(), dataset['Age'].max(), dataset['Age'].median()
```

Out[14]:
```
(0.42, 80.0, 28.0)
```

In [16]:
```
# So in above 2 rows, min. age account for 0% percentile and max. age accounts for
# and median age is 50% percentile of age
```

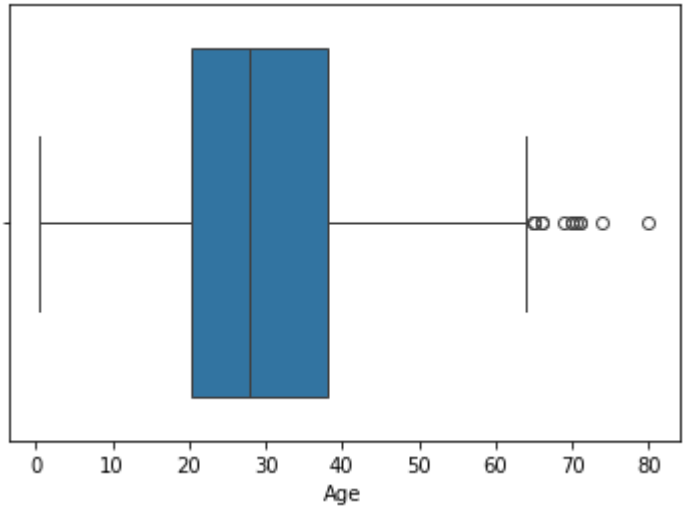
In [17]:
```
dataset.describe()
```

Out[17]:

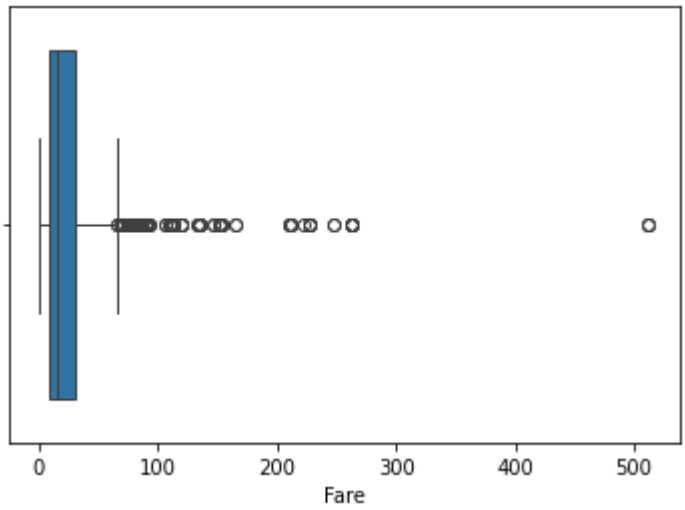
	Survived	Pclass	Age	Siblings/Spouses Aboard	Parents/Children Aboard	Fare
count	887.000000	887.000000	887.000000	887.000000	887.000000	887.00000
mean	0.385569	2.305524	29.471443	0.525366	0.383315	32.30542
std	0.487004	0.836662	14.121908	1.104669	0.807466	49.78204
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.00000
25%	0.000000	2.000000	20.250000	0.000000	0.000000	7.92500
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.45420
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.13750
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.32920

In [20]:
```
#If you see closely on age you can see that
# min(0%)      : 0.42
# Q1 : 25%     : 20.25
# Q2 : 50%     : 28.00
# Q3 : 75%     : 38.00
# Q4 : max(80%): 80.00
# So you can see the huge difference between Q3 and Q4. So it is clear that outlier
# Also difference between min (0%) and Q1 is significant larger, so there is also c
# median (Q2) is 28, so it is evident that the median is inclined towards left side
# So this whole analysis tell that there is definitely outlier present in this data
```

In [23]:
```
# To show it in the boxplot
sns.boxplot(x='Age', data=dataset)
plt.show()
```



In [25]:
```
# To show it in the boxplot
sns.boxplot(x='Fare', data=dataset)
plt.show()
```



4_Measures of Shape - Skewness

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
dataset = pd.read_csv('titanic.csv')
```

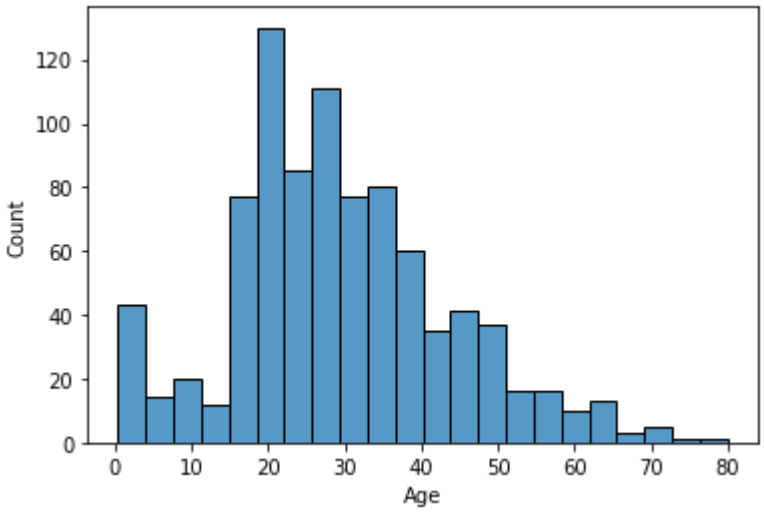
In [4]:
```
dataset.head(3)
```

Out[4]:

	Survived	Pclass	Name	Sex	Age	Siblings/Spouses Aboard	Parents/Children Aboard	Fare
0	0	3	Mr. Owen Harris Braund	male	22.0	1	0	7.2500
1	1	1	Mrs. John Bradley (Florence Briggs Thayer) Cum...	female	38.0	1	0	71.2833
2	1	3	Miss. Laina Heikkinen	female	26.0	0	0	7.9250

To see if Age has skewness or no skewness

In [6]:
```
sns.histplot(x='Age', data=dataset)
plt.show()
```



This is right skewed chart (Positive skewness)

In [8]:
```
# If skew is greater than zero - Positive skewness and vice versa
dataset['Age'].skew()
```

Out[8]:
```
0.44718857190799916
```

5_Probability - Correlation

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
dataset = pd.read_csv('tips.csv')
```

In [3]:
```
dataset.head(3)
```

Out[3]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3

In [4]:
```
dataset.isnull().sum()
```

Out[4]:
```
total_bill    0
tip           0
sex           0
smoker        0
day           0
time          0
size          0
dtype: int64
```

In [5]:
```
# To check datatypes in dataset
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   total_bill  244 non-null    float64
 1   tip         244 non-null    float64
 2   sex         244 non-null    object 
 3   smoker      244 non-null    object 
 4   day         244 non-null    object 
 5   time        244 non-null    object 
 6   size        244 non-null    int64  
dtypes: float64(2), int64(1), object(4)
memory usage: 13.5+ KB
```

In [6]:
```
dataset.select_dtypes("float64" ,"int64")
```

Out[6]:

	total_bill	tip
0	16.99	1.01
1	10.34	1.66
2	21.01	3.50
3	23.68	3.31
4	24.59	3.61
...	...	...
239	29.03	5.92
240	27.18	2.00
241	22.67	2.00
242	17.82	1.75
243	18.78	3.00

244 rows × 2 columns

In [10]:
```
data_cor = dataset.select_dtypes("float64" ,"int64").corr()
data_cor
```

Out[10]:

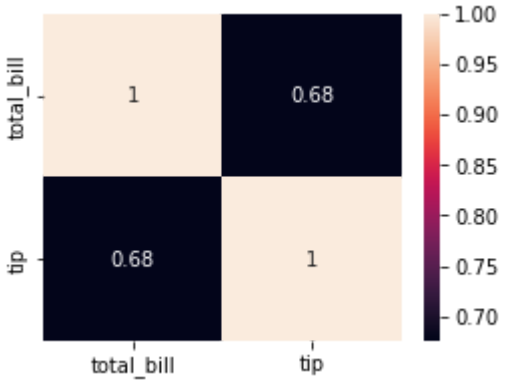
	total_bill	tip
total_bill	1.000000	0.675734
tip	0.675734	1.000000

In [11]:
```
data_cov = dataset.select_dtypes("float64" ,"int64").cov()
data_cov
```

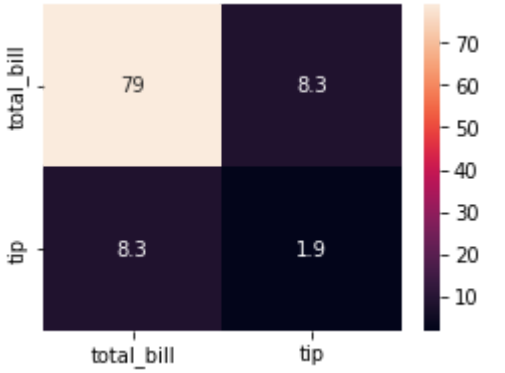
Out[11]:

	total_bill	tip
total_bill	79.252939	8.323502
tip	8.323502	1.914455

In [16]:
```
plt.figure(figsize=(4,3))
sns.heatmap(data_cor, annot=True)
plt.show()
```



In [17]:
```
plt.figure(figsize=(4,3))
sns.heatmap(data_cov, annot=True)
plt.show()
```



In [ ]:

6_Central Limit Theorem

In [40]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [41]:
```
# Generate random data by using list comprehension

pop_data = [np.random.randint(10,100) for i in range(10000)]
#pop_data
```

In [42]:
```
# the above line of code could be written as:
pop_data = []
for i in range(10000):
    pop_data.append(np.random.randint(10,100))
#pop_data
```

In [43]:
```
len(pop_data)
```

Out[43]: 10000

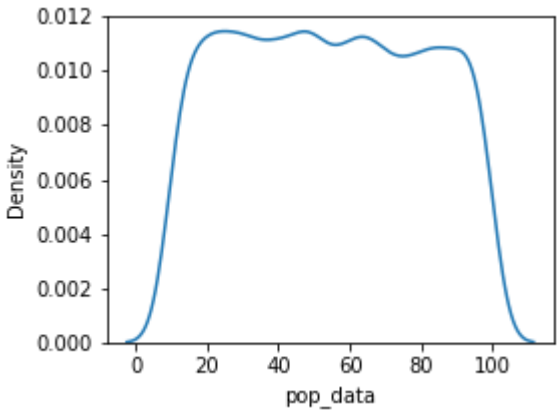
In [44]:
```
# TO convert population data into a csv file
pop_table = pd.DataFrame({'pop_data':pop_data})
pop_table
```

Out[44]:

	pop_data
0	55
1	22
2	87
3	46
4	29
...	...
9995	51
9996	60
9997	42
9998	99
9999	48

10000 rows × 1 columns

In [45]:
```
plt.figure(figsize=(4,3))
sns.kdeplot(x='pop_data', data=pop_table)
plt.show()
```



above graph shows that our data is not normally distributed, so we will apply CLT

In [46]:
```
# First we will pick up random samples from population data
# Pre-req: Sample should not be more than 10% population and more than 30 samples s

# so calculate 10% of 10000 data

10/100 * 10000
```

Out[46]:
```
1000.0
```

That means i.e. n>30 and n< 1000, so are taking n= [50,500]

In [47]:
```
# To pick random data from population data
np.random.choice(pop_data)
```

Out[47]:
```
82
```

In [ ]:
```
# So will take sample data less than 1000
sample_mean = []
# to take number of sample data 50 (to meet requirement n>30)
for no_of_sample in range(50):
    sample_data = []
    # to take number of sample data less than 1000 (so will take 500 sample)
    for i in range(500):
        sample_data.append(np.random.choice(pop_data))
    # To calculate mean of sample data
    sample_mean.append(np.mean(sample_data))
```

In [ ]:
```
len(sample_data), len(sample_mean)
```

In [ ]:
```
sample_data
```

In [ ]:
```
sample_mean
```

In [ ]:
```
# To see data in sample_mean is normally distributed or not
sample_mean_DF = pd.DataFrame({"Sample_mean":sample_mean})
```

In [ ]:
```
sample_mean_DF
```

In [ ]:
```
plt.figure(figsize=(4,5))
sns.kdeplot(x="Sample_mean", data=sample_mean_DF)
plt.show()
```

So the data is normally distributed

In [ ]:
```
# To meat another requirement of CLT that is the mean of population data and the me
# so we will check the both means
np.mean(pop_data), np.mean(sample_mean)
```

7_Calculating Z-test

In [11]:
```
import scipy.stats as st
import numpy as np
```

In [19]:
```
# To calculate Z-value (from Z-table)
z_table = st.norm.ppf(1-0.05)
z_table
```

Out[19]: 1.6448536269514722

In [20]:
```
s_x = 90
p_u = 82
p_std = 20
n = 81
```

In [21]:
```
z_cal = (s_x - p_u) / (p_std/np.sqrt(n))
z_cal
```

Out[21]: 3.5999999999999996

In [24]:
```
if z_table < z_cal:
    print("Alternate Hypothesis (Ha) is correct")
else:
    print("Null hypothesis (Ho) is correct")
```

```
Alternate Hypothesis (Ha) is correct
```

In [ ]:

8_Calculating T-test

In [17]:
```
import scipy.stats as st
import numpy as np
```

In [18]:
```
Ho = "Weight of bag is 150gm"
Ha = "Weight of bag is less than 150gm"
```

In [19]:
```
t_table = st.t.ppf(0.05,24)
t_table
```

Out[19]: -1.7108820799094282

In [20]:
```
u_p = 150
x_s = 148
n_s = 25
std_s = 5
```

In [21]:
```
t_cal = (x_s - u_p)/(std_s/np.sqrt(n_s))
t_cal
```

Out[21]: -2.0

In [22]:
```
if t_table > t_cal:
    print(Ha)
else:
    print(Ho)
```

```
Weight of bag is less than 150gm
```

In [ ]:

9_Calculating Chi-Square Test

9.1_To check goodness of data

In [1]:
```
import numpy as np
```

In [4]:
```
ob = np.array([22,17,20,26,22,13])
ex = np.array([20,20,20,20,20,20])
```

In [5]:
```
ob-ex
```

Out[5]:
```
array([ 2, -3,  0,  6,  2, -7])
```

In [9]:
```
np.sum(np.square(ob-ex)/ex)
```

Out[9]:
```
5.1000000000000005
```

9.2_To check dependency of variables

In [22]:
```
row1 = np.array([40,45,25,10])
row2 = np.array([35,30,20,30])
```

In [25]:
```
sum_r1 = np.sum(row1)
sum_r2 = np.sum(row2)
sum_row = np.array([sum_r1, sum_r2])
sum_row
```

Out[25]:
```
array([120, 115])
```

In [26]:
```
sum_col = row1 + row2
sum_col
```

Out[26]:
```
array([75, 75, 45, 40])
```

In [32]:
```
exp = []
for i in sum_row:
    for j in sum_col:
        exp.append(i*j/235)
print(exp)
```

```
[38.297872340425535, 38.297872340425535, 22.97872340425532, 20.425531914893618, 36.7
02127659574465, 36.702127659574465, 22.02127659574468, 19.574468085106382]
```

In [34]:
```
# join both columns for observed values
obj = np.array([40,45,25,10,35,30,20,30])
```

In [36]:
```
np.sum(np.square(obj - exp)/exp)
```

Out[36]: 13.788747987117553

In [ ]: