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

In [2]:
```python
# Q1. Generate a list of 100 integers containing values between 90 to R30 and store it in the variable `int_lis
# After generating the list, find the following:
# A. Write a Python function to calculate the mean of a given list of numbers.Create a function to find the med
data=np.random.randint(90,130,100 )
def calculate_mean(x):
    return np.mean(x)
def calculate_median(x):
    return np.median(x)
print("The mean of Data is :",calculate_mean(data))
print("The median of Data is :",calculate_median(data))
```

```
The mean of Data is : 111.73
The median of Data is : 114.0
```

In [3]:
```python
# B. Develop a program to compute the mode of a list of integers.
def calculate_mode(x):
    return statistics.mode(x)
print("The Mode of Data is:",calculate_mode(data))
```

```
The Mode of Data is: 117
```

In [4]:
```python
# C. Implement a function to calculate the weighted mean of a list of values and their corresponding weights.
values=np.random.rand(1,100)
value=values[0]
sum1=0
for i in range(0,100):
    sum1=sum1+ value[i]*data[i]
sum_of_value=np.sum(value)
weighted_mean = sum1/sum_of_value
print("The Value of Weighted mean :",weighted_mean)
```

```
The Value of Weighted mean : 111.046731422591
```

In [5]:
```python
# D.  Write a Python function to find the geometric mean of a list of positive numbers.
def geometric_mean(x):
    n= len(x)
    product=1
    for i in x:
        product=product*i
    value=product**(1/n)
    return value
result= geometric_mean(value)
print("The geometric mean is:",result)
```

```
The geometric mean is: 0.3602062855831996
```

In [6]:
```python
# E. Create a program to calculate the harmonic mean of a list of values.
def harmonic_mean(x):
    n=len(x)
    sum1=0
    for element in x:
        sum1=sum1 + (1/element)

    value= n/sum1
    return value
print("The Value of Harmonic mean is :",harmonic_mean(value))
```

```
The Value of Harmonic mean is : 0.1543364057814751
```

In [7]:
```python
# F. Build a function to determine the midrange of a list of numbers (average of the minimum and maximum).
def find_midvalue(x):
    maxi=np.max(x)
    mini=np.min(x)
    avg=(maxi+ mini)/2
    return avg
val= np.random.randint(90,130,100)
print("The Avg of Minimum and Maximum value is :",find_midvalue(val))
```

```
The Avg of Minimum and Maximum value is : 110.0
```

In [8]:
```python
# G.Implement a Python program to find the trimmed mean of a list, excluding a certain percentage of  outliers.
data=list(data)
data.append(-10)
data.append(-20)
```

```
data.append(500)
data.append(700)
data.append(1000)
print(data)
```

```
[114, 117, 122, 125, 114, 112, 105, 105, 120, 117, 119, 124, 111, 128, 90, 120, 123, 120, 110, 102, 105, 123, 1
20, 123, 102, 113, 118, 90, 122, 107, 117, 90, 93, 97, 124, 127, 108, 99, 128, 99, 128, 94, 105, 119, 107, 124,
98, 116, 112, 109, 127, 114, 116, 112, 119, 104, 116, 120, 118, 117, 117, 115, 124, 120, 125, 114, 100, 127, 98
, 116, 94, 126, 92, 114, 107, 121, 95, 110, 96, 91, 125, 108, 91, 109, 107, 101, 121, 112, 97, 126, 118, 110, 9
1, 121, 96, 116, 108, 118, 101, 117, -10, -20, 500, 700, 1000]
```

In [9]:
```
data=sorted(data)
print(data)
```

```
[-20, -10, 90, 90, 90, 91, 91, 91, 92, 93, 94, 94, 95, 96, 96, 97, 97, 98, 98, 99, 99, 100, 101, 101, 102, 102,
104, 105, 105, 105, 105, 107, 107, 107, 107, 108, 108, 108, 109, 109, 110, 110, 110, 111, 112, 112, 112, 112, 1
13, 114, 114, 114, 114, 114, 115, 116, 116, 116, 116, 116, 117, 117, 117, 117, 117, 117, 118, 118, 118, 118, 11
9, 119, 119, 120, 120, 120, 120, 120, 120, 121, 121, 121, 122, 122, 123, 123, 123, 124, 124, 124, 124, 125, 125
, 125, 126, 126, 127, 127, 127, 128, 128, 128, 500, 700, 1000]
```

In [10]:
```
ls=np.percentile(data,[25,75])
Q1=ls[0]
Q3=ls[1]
print(Q1)
print(Q3)
```

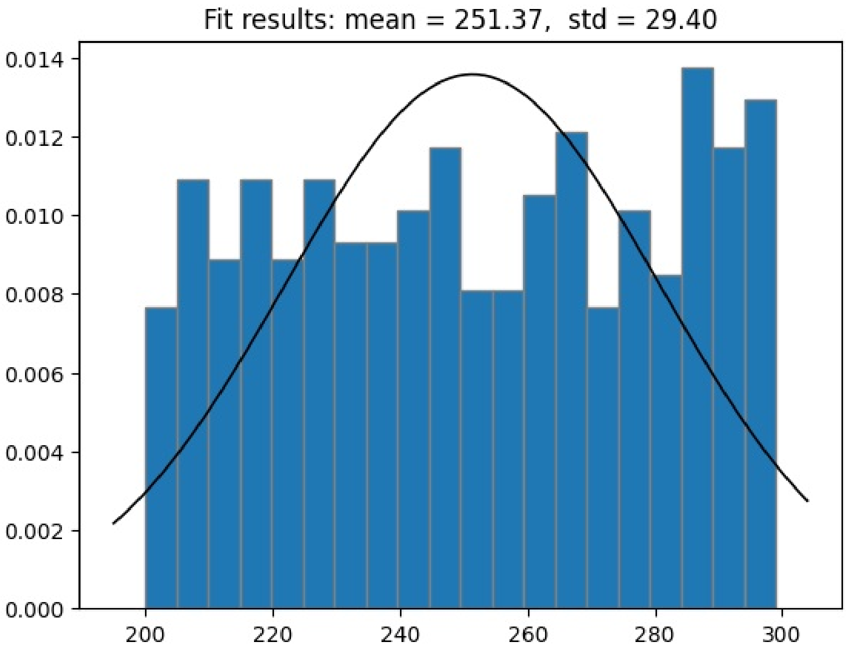
```
104.0
120.0
```

In [11]:
```
IQR=Q3-Q1
lf=Q1-IQR*1.5
new_data=[]
uf=Q3+1.5*IQR
for dat in data:
    if(dat<=uf and dat>=lf):
        new_data.append(dat)
print("The Trimmed mean of this Data is after removing outlier :",np.mean(new_data))
```

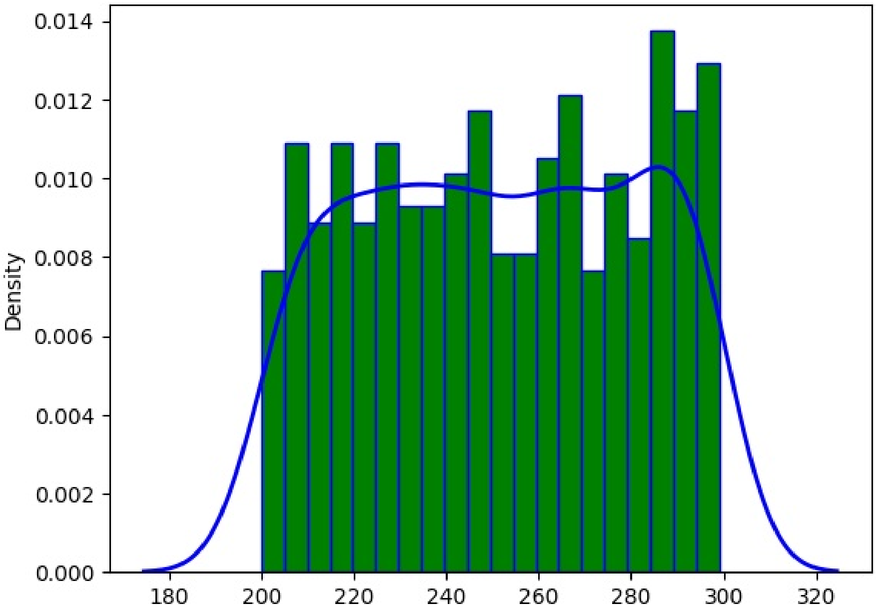
```
The Trimmed mean of this Data is after removing outlier : 111.73
```

In [12]:
```
# 2. Generate a list of 500 integers containing values between 200 to 300 and store it in the variable `int_lis
 # After generating the list, find the following:
```

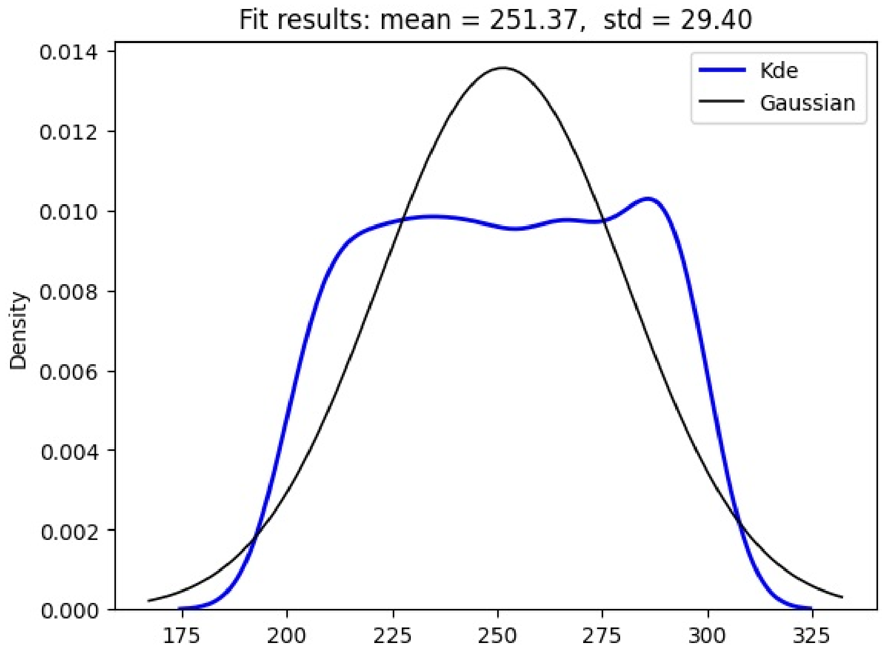
In [13]:
```
#  (i) Compare the given list of visualization for the given data:
#     1. Frequency & Gaussian distribution

# For Frequency distribution histogram
data1=np.random.randint(200,300,500)
plt.hist(data1,bins=20,edgecolor="grey",density=True)

# for Gaussian distribution
mean=np.mean(data1)
std_dev= np.std(data1)
xmin,xmax= plt.xlim() # -> This give the min x value and max x value
x=np.linspace(xmin,xmax,100)
p=stats.norm.pdf(x,mean,std_dev)
plt.plot(x,p,'k',linewidth=1.2)
title= "Fit results: mean = %.2f,  std = %.2f" % (mean, std_dev)

plt.title(title)
plt.show()
```



In [14]:
```
#     2. Frequency smoothened KDE plot
plt.hist(data1,bins=20, density=True,edgecolor="blue",color="g")
sns.kdeplot(data1,color='blue',linewidth=2)
plt.show()
```



In [15]:
```
#     3. Gaussian distribution & smoothened KDE plot

sns.kdeplot(data1,color='blue',linewidth=2,label='Kde')
# for Gaussian distribution
mean=np.mean(data1)
std_dev= np.std(data1)
xmin,xmax= plt.xlim() # -> This give the min x value and max x value
x=np.linspace(xmin,xmax,100)
p=stats.norm.pdf(x,mean,std_dev)
plt.plot(x,p,'k',linewidth=1.2,label='Gaussian')
title= "Fit results: mean = %.2f,  std = %.2f" % (mean, std_dev)

plt.title(title)
plt.legend()
plt.show()
```



In [16]:
```
#   (ii) Write a Python function to calculate the range of a given list of numbers.
def finding_range(x):
    max=str(np.max(x))
    min=str(np.min(x))
    return f' from {min} to {max}'
print("The Range of data is :",finding_range(data1))
```

```
The Range of data is :  from 200 to 299
```

In [17]:
```
#   (iii) Create a program to find the variance and standard deviation of a list of numbers.
def finding_var_std(x):
    var=str(np.var(x))
    std_dev=str(np.std(x))
    return f'Variance is :{var} and standard variance is :{std_dev}'
print(finding_var_std(data1))
```

```
Variance is :864.3965759999999 and standard variance is :29.40062203423594
```

In [18]:
```
# (iv) Implement a function to compute the interquartile range (IQR) of a list of values.
def finding_IQR(x):
    lis=np.percentile(x,[25,75])
    Q1=lis[0]
    Q3= lis[1]
    IQR= Q3-Q1
    return IQR
print("The interquartile range (IQR) is :",finding_IQR(data1))
```

```
The interquartile range (IQR) is : 50.0
```

In [19]:
```
#   (v) Build a program to calculate the coefficient of variation for a dataset.
# Coefficient of Variation:- The Ratio between standard deviation and absolute mean.
def finding_coeff_vari(x):
    std_dev= np.std(x)
    mean=abs(np.mean(x))
    ratio= std_dev/mean
    return ratio
print("The Coefficient of Data is :",finding_coeff_vari(data1))
```

```
The Coefficient of Data is : 0.11696246950381886
```

In [20]:
```
#   (vi) Write a Python function to find the mean absolute deviation (MAD) of a list of numbers.
def finding_mean_abso_devi(x):
    mean= np.mean(data1)
    sum1=0
    n= len(data1)
    for data in data1:
        sum1 = sum1 + abs(data-mean)
    value= sum1/n
    return value
print("The Mean absolute deviation(MOD) of data1 is :",finding_mean_abso_devi(data1))
```

```
The Mean absolute deviation(MOD) of data1 is : 25.579999999999999
```

In [21]:
```
#   (vii) Create a program to calculate the quartile deviation of a list of values.
# Quartile deviation :- It is the average of diffenece between the first quartile and third quartile in
# the frequency distribution table.
```

```
def finding_quartile_devi(x):
    lis=np.percentile(x,[25,75])
    Q1=lis[0]
    Q3= lis[1]
    devi= (Q3-Q1)/2
    return devi
print("The Quartile deviation of the Data is :",finding_quartile_devi(data1))
```

```
The Quartile deviation of the Data is : 25.0
```

In [22]:
```
# (viii) Implement a function to find the range-based coefficient of dispersion for a dataset.
# Range-Based-Coefficient of Deviation :- the Ratio of (maximum value- minimum value) and (maximum value + mini
def finding_range_based_coeff_dispersion(x):
    maxi= np.max(x)
    mini= np.min(x)
    num= maxi-mini
    denomi= maxi+mini
    value= num/denomi
    return value
print("The Range -Based-Coefficient of Deviation is :",finding_range_based_coeff_dispersion(data1))
```

```
The Range -Based-Coefficient of Deviation is : 0.19839679358717435
```

In [23]:
```
# Q3. Write a Python class representing a discrete random variable with methods to calculate its expected
# value and variance.
class RandomVariable:
    def __init__(self,values,probabilities):
        self.values=values
        self.probabilities = probabilities
    def finding_expected_value(self):
        leng= len(self.values)
        sum1=0
        for i in range(0,leng):
            sum1 =sum1 + self.values[i]*self.probabilities[i]
        return sum1
    def finding_variance(self):
        leng= len(self.values)
        sum2=0
        sum1=0
        for i in range(0,leng):
            sum2 =sum2 + (self.values[i]**2)*self.probabilities[i]
            sum1 = sum1 +(self.values[i])*self.probabilities[i]
        dif= sum2-sum1
        return dif
value=[1,2,3,4,5,6]
prob=[1/6 for i in range(0,len(value))]
obj1= RandomVariable(value,prob)
print("The Expected Value is:",obj1.finding_expected_value())
print("The Variance is :",obj1.finding_variance())
```

```
The Expected Value is: 3.5
The Variance is : 11.666666666666666
```

In [24]:
```
# Q4.  Implement a program to simulate the rolling of a fair six-sided die and calculate the expected value and
# variance of the outcomes.
no_die_sided=[1,2,3,4,5,6]
data=np.random.choice(no_die_sided,20)
probability_of_data=[]
for outcome in no_die_sided:
    count=0
    n= len(data)
    for dat in data:
        if outcome==dat:
            count+=1
    probability_of_data.append(count/n)
print(probability_of_data)
def finding_expected_value(no_die_sided,probability_of_data):
        leng= len(no_die_sided)
        sum1=0
        for i in range(0,leng):
            sum1 =sum1 + no_die_sided[i]*probability_of_data[i]
        return sum1
def finding_variance_value(no_die_sided,probability_of_data):
        leng= len(no_die_sided)
        sum2=0
        sum1=0
        for i in range(0,leng):
            sum2 =sum2 + (no_die_sided[i]**2)*probability_of_data[i]
            sum1 = sum1 +(no_die_sided[i])*probability_of_data[i]
        dif= sum2-sum1
        return dif

print("The Expected Value is:",finding_expected_value(no_die_sided,probability_of_data))
print("The Variance is :",finding_variance_value(no_die_sided,probability_of_data))
```

```
[0.2, 0.15, 0.1, 0.15, 0.1, 0.3]
The Expected Value is: 3.6999999999999997
The Variance is : 13.7
```

In [25]:

```python
# Q5. Create a Python function to generate random samples from a given probability distribution (e.g.,
# binomial, Poisson) and calculate their mean and variance.

# for Binomial Distribution-->
n=10 # No of trial
p=0.5  # Probability of success
size=1000  # size of data
data=np.random.binomial(n,p,size)
sample=np.random.choice(data,50)
print("The mean of sample of binomial distribution is :",np.mean(sample))
print("The variance of sample of binomial distribution is :",np.var(sample))

# for Poisson Distribution
lamda= 10
size=1000
data= np.random.poisson(lamda,size)
sample = np.random.choice(data,60)
print("The mean of sample of poisson distribution is :",np.mean(sample))
print("The variance of sample of poisson distribution is :",np.var(sample))
```

```
The mean of sample of binomial distribution is : 5.2
The variance of sample of binomial distribution is : 2.4000000000000004
The mean of sample of poisson distribution is : 9.5
The variance of sample of poisson distribution is : 10.05
```

In [26]:

```python
# Q6.Write a Python script to generate random numbers from a Gaussian (normal) distribution and compute
# the mean, variance, and standard deviation of the samples.

def finding_m_var_std_dev_normal_dis(m,std_d,size):
    data= np.random.normal(m,std_d,size)
    sample= np.random.choice(data,110)
    print("The Mean of sample is :",np.mean(sample))
    print("The variance of sample is :",np.var(sample))
    print("The std_deviation of sample is :",np.std(sample))
mean=0
std_dev=1
size=2000
finding_m_var_std_dev_normal_dis(mean,std_dev,size)
```

```
The Mean of sample is : 0.09012138773569484
The variance of sample is : 0.9568512983656456
The std_deviation of sample is : 0.9781877623266637
```

In [27]:

```python
# Q7. Use seaborn library to load tips dataset. Find the following from the dataset for the columns total_bill
# and tip`:



#   (i) Write a Python function that calculates their skewness.


#   (ii) Create a program that determines whether the columns exhibit positive skewness, negative skewness, or .
# approximately symmetric.


#   (iii) Write a function that calculates the covariance between two columns.


#   (iv) Implement a Python program that calculates the Pearson correlation coefficient between two columns.


#   (v) Write a script to visualize the correlation between two specific columns in a Pandas DataFrame using
# scatter p
```

In [28]:

```python
df=sns.load_dataset('tips')
df
```

Out[28]:

	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
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

In [29]:
```
#   (i) Write a Python function that calculates their skewness.

sk=stats.skew(df['total_bill'])
print(sk)
sk=stats.skew(df['tip'])
print(sk)
print("Both the columns are highly skewed")
```

```
1.1262346334818638
1.4564266884221506
Both the columns are highly skewed
```

In [30]:
```
#   (ii) Create a program that determines whether the columns exhibit positive skewness, negative skewness, or 
# approximately symmetric.
def determin_skew_nature(data):
    sk=stats.skew(data)
    if(sk <= 0.5 and sk >= -0.5):
        print("Distribution is almost symmetric")
    elif(sk<-0.5):
        print("Negative skewness")
    else:
        print("Highly positive skewness")
determin_skew_nature(df['total_bill'])
determin_skew_nature(df['tip'])
```

```
Highly positive skewness
Highly positive skewness
```

In [31]:
```
#   (iii) Write a function that calculates the covariance between two columns.
data=df[['total_bill','tip']]
data.cov()
```

Out[31]:

	total_bill	tip
total_bill	79.252939	8.323502
tip	8.323502	1.914455

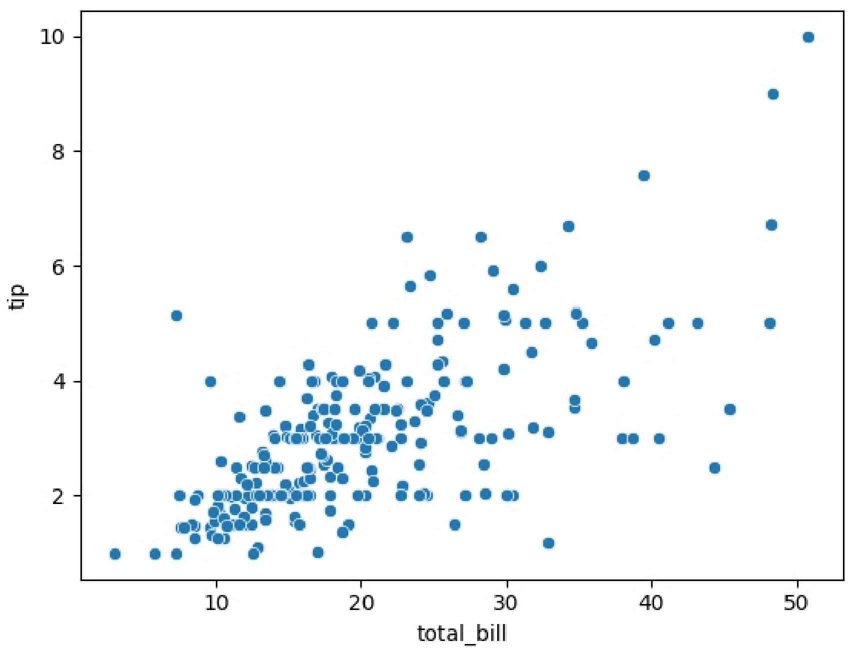
In [32]:
```
#   (iv) Implement a Python program that calculates the Pearson correlation coefficient between two columns.
data.corr()
```

Out[32]:

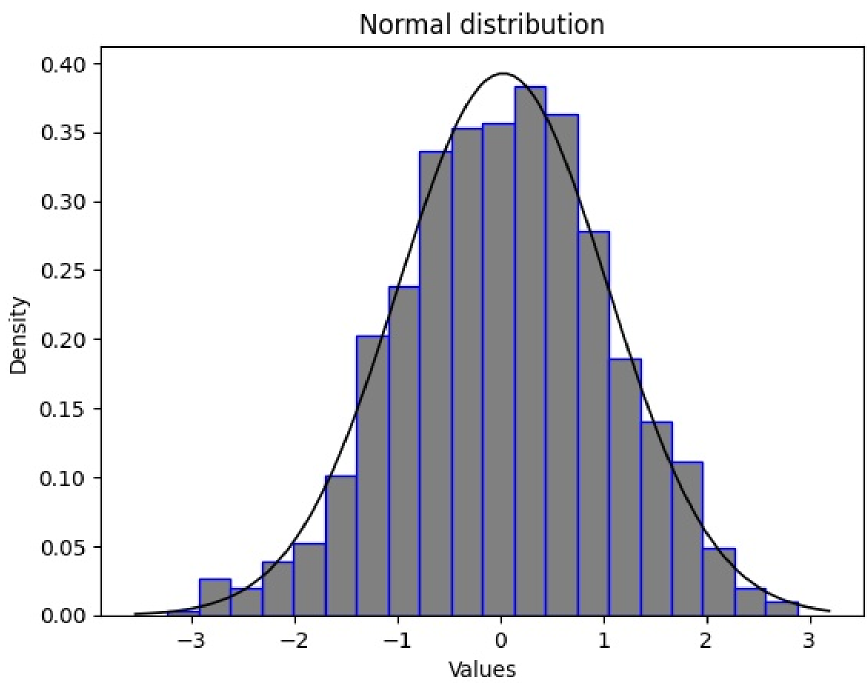
	total_bill	tip
total_bill	1.000000	0.675734
tip	0.675734	1.000000

In [33]:
```
#   (v) Write a script to visualize the correlation between two specific columns in a Pandas DataFrame using
# scatter p.
sns.scatterplot(x=data.total_bill,y=data.tip,data=data)
```

Out[33]: <Axes: xlabel='total_bill', ylabel='tip'>

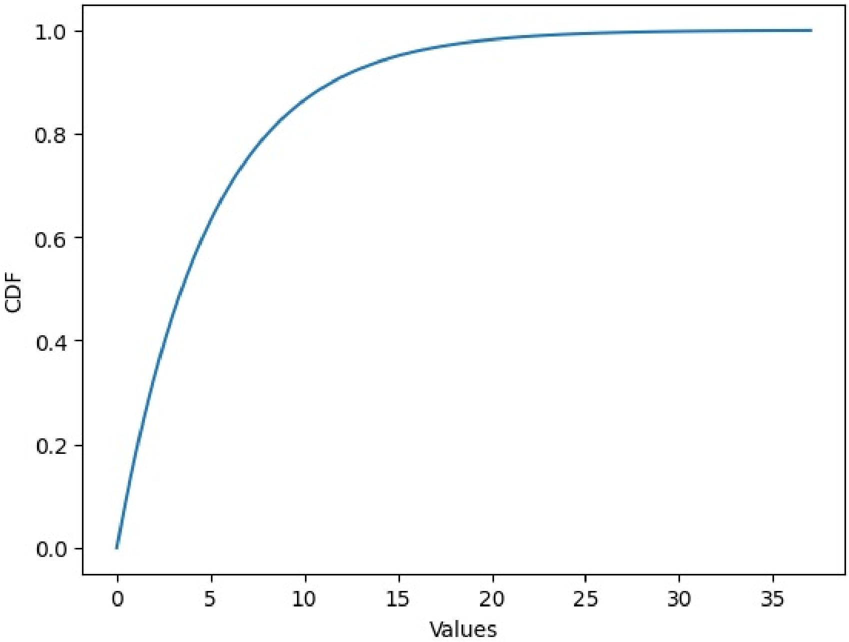


In [34]:
```
# Q8. Write a Python function to calculate the probability density function (PDF) of a continuous random
# variable for a given normal distribution.
data= np.random.normal(0,1,1000)
plt.hist(data,bins=20, edgecolor='b',color='grey',density=True)
mean=np.mean(data)
std_dev= np.std(data)
xmin,xmax=plt.xlim()
x=np.linspace(xmin,xmax,100)
p=stats.norm.pdf(x,mean,std_dev)
plt.plot(x,p,'k',linewidth=1.2)
plt.title('Normal distribution')
plt.xlabel('Values')
plt.ylabel('Density')
plt.show()
```



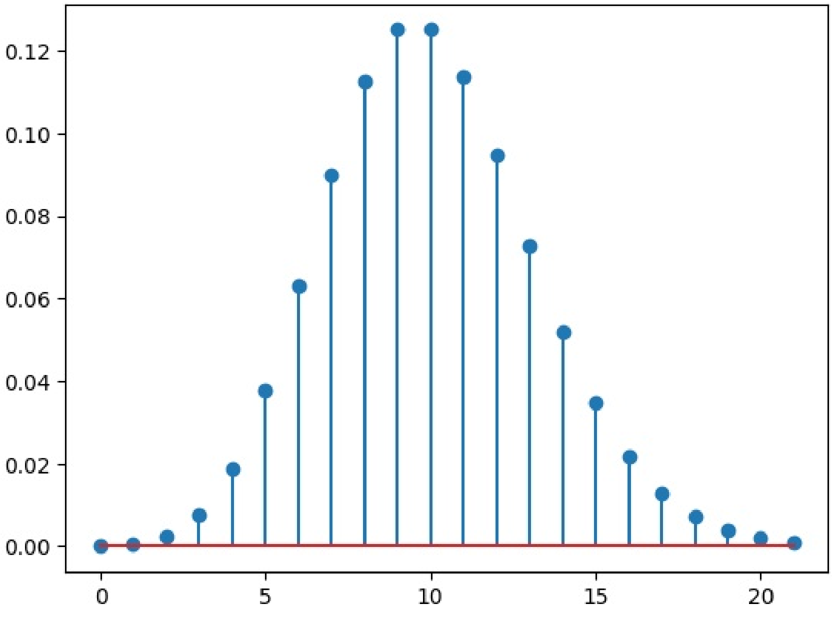
In [35]:
```
# Q9.Create a program to calculate the cumulative distribution function (CDF) of exponential distribution.
scale=5
size=1000
data= np.random.exponential(scale,size)
x = np.linspace(0, np.max(data), 100)
cdf = stats.expon.cdf(x, scale=scale)
plt.plot(x,cdf)
plt.xlabel('Values')
```

```
plt.ylabel('CDF')
plt.show()
```



In [36]:
```
# Q10. Write a Python function to calculate the probability mass function (PMF) of Poisson distribution.
lamda=10
size=1000
data=np.random.poisson(lamda,size)
    # Calculate the PMF values
x = np.arange(0, np.max(data) + 1)
pmf = stats.poisson.pmf(x, lamda)

    # Plot the PMF
plt.stem(x, pmf)
```

Out[36]:
```
<StemContainer object of 3 artists>
```



In [50]:
```
# Q11. A company wants to test if a new website layout leads to a higher conversion rate (percentage of visitors
# who make a purchase). They collect data from the old and new layouts to compare.
import numpy as np
from statsmodels.stats.proportion import proportions_ztest
null_hypothesis ="The new website layout does not lead to a higher conversion rate compared to the old layout."
alternate_hypothesis ="The new website layout leads to a higher conversion rate compared to the old layout."
old_layout = np.array([1] * 50 + [0] * 950)
new_layout = np.array([1] * 70 + [0] * 930)
conv_old= np.sum(old_layout)
con_old = np.sum(new_layout)
no_old_layout= len(old_layout)
no_new_layout= len(new_layout)
z_statistics,p_value= proportions_ztest([conv_old,conv_new],[no_old_layout,no_new_layout])
print("The value of Z statistics is :",z_statistics)
print("The value of p value is :",p_value)
alpha=0.05
```

```
if p_value<alpha :
    print("Reject the null_hypothesis")
else:
    print("Fail to reject the null_hypothesis")
```

```
The value of Z statistics is : -1.883108942886774
The value of p value is : 0.05968560553242621
Fail to reject the null_hypothesis
```

In [52]:

```
# Q12 A tutoring service claims that its program improves students' exam scores. A sample of students who
# participated in the program was taken, and their scores beNore and aNter the program were recorded.
# Use z-test to find if the claims made by tutor are true or false.
before_program = np.array([75, 80, 85, 70, 90, 78, 92, 88, 82, 87])
after_program = np.array([80, 85, 90, 80, 92, 80, 95, 90, 85, 88])
from statsmodels.stats.weightstats import ztest

null_hypothesis= " the mean exam scores before and after the program are the same ."
alternate_hypothesis = "the mean exam score after the program is higher than the mean exam score before the prog

zscore,p_value=ztest(before_program,after_program,alternative='larger')
print("The value of Zscore is :",zscore)
print("The value of p_value is :",p_value)
alpha = 0.05
if p_value >alpha :
    print("fail to reject the null hypothesis")
else:
    print("Reject null hypothesis")
```

```
The value of Zscore is : -1.3600371723457605
The value of p_value is : 0.9130909194908616
fail to reject the null hypothesis
```

In [53]:

```
# Q13. A pharmaceutical company wants to determine if a new drug is effective in reducing blood pressure. They
# conduct a study and record blood pressure measurements before and after administering the drug.
before_drug = np.array([145, 150, 140, 135, 155, 160, 152, 148, 130, 138])
after_drug = np.array([130, 140, 132, 128, 145, 148, 138, 136, 125, 130])
import numpy as np
from scipy import stats
differences = after_drug - before_drug
Z_statistic = differences.mean() / (differences.std(ddof=1) / np.sqrt(len(differences)))
p_value = 2 * (1 - stats.norm.cdf(np.abs(Z_statistic)))  # two-tailed test

print(f"Z-statistic: {Z_statistic:.4f}")
print(f"P-value: {p_value:.4f}")


alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: The drug is effective in reducing blood pressure.")
else:
    print("Fail to reject the null hypothesis: There is no significant evidence that the drug reduces blood pres
```

```
Z-statistic: -10.0499
P-value: 0.0000
Reject the null hypothesis: The drug is effective in reducing blood pressure.
```

In [54]:

```
# Q14. A customer service department claims that their average response time is less than 5 minutes. A sample
# of recent customer interactions was taken, and the response times were recorded.

response_times = np.array([4.3, 3.8, 5.1, 4.9, 4.7, 4.2, 5.2, 4.5, 4.6, 4.4])
import numpy as np
from scipy import stats

sample_mean = np.mean(response_times)
sample_std = np.std(response_times, ddof=1)

population_mean = 5


Z_statistic = (sample_mean - population_mean) / (sample_std / np.sqrt(len(response_times)))


p_value = stats.norm.cdf(Z_statistic)

print(f"Z-statistic: {Z_statistic:.4f}")
print(f"P-value: {p_value:.4f}")
alpha = 0.05
if p_value < alpha:
    print("\nReject the null hypothesis: The average response time is less than 5 minutes.")
else:
    print("\nFail to reject the null hypothesis: There is no significant evidence that the average response time
```

```
Z-statistic: -3.1845
P-value: 0.0007

Reject the null hypothesis: The average response time is less than 5 minutes.
```

In [40]:

```
#Q15.  A company is testing two different website layouts to see which one leads to higher click-through ratesV
# Write a Python function to perform an A/B test analysis, including calculating the t-statistic, degrees of
# freedom, and p-value.
layout_a_clicks = [28, 32, 33, 29, 31, 34, 30, 35, 36, 37]
layout_b_clicks = [40, 41, 38, 42, 39, 44, 43, 41, 45, 47]
def AB_test_analysis(layout_a_clicks,layout_b_clicks):
    mean_A=sum(layout_a_clicks)/len(layout_a_clicks)
    mean_B=sum(layout_b_clicks)/len(layout_b_clicks)
    null_hypothesis="The mean click-through rate for website layout A is equal to the mean click-through rate fo
    alternate_hypothesis= "The mean click-through rate for website layout A is not equal to the mean click-throu
    t_statistics,p_value= stats.ttest_ind(layout_a_clicks,layout_b_clicks)
    n1 = len(layout_a_clicks)
    n2 = len(layout_b_clicks)
    degree_of_freedom = n1+n2-2
    alpha=0.05

    if p_value > alpha:
        print("Fail to reject Null_hypothesis")
    else:
        print("Reject the null_hypothesis")


    print("T-statistics value is :",t_statistics)
    print("The Degree of freedom value is :",degree_of_freedom)
    print("P_value of this analysis is:",p_value)
AB_test_analysis(layout_a_clicks,layout_b_clicks)
```

```
Reject the null_hypothesis
T-statistics value is : -7.298102156175071
The Degree of freedom value is : 18
P_value of this analysis is: 8.833437608301987e-07
```

In [41]:

```
# Q16. A pharmaceutical company wants to deterine if a new drug is more effective than an existing drug in
# reducing cholesterol levels. Create a program to analyze the clinical trial data and calculate the t_statisti
existing_drug_levels = [180, 182, 175, 185, 178, 176, 172, 184, 179, 183]
new_drug_levels = [170, 172, 165, 168, 175, 173, 170, 178, 172, 176]
def analysis_clinical_trial(existing_drug_levels,new_drug_levels):
    mean_exist= sum(existing_drug_levels)/len(existing_drug_levels)
    mean_new = sum(new_drug_levels)/len(new_drug_levels)
    null_hypothesis="The mean reduction in cholesterol levels with the new drug is equal to the mean reduction i
    alternate_hypothesis="The mean reduction in cholesterol levels with the new drug is different from the mean
    t_statistics,p_value=stats.ttest_ind(existing_drug_levels,new_drug_levels)
    n1= len(existing_drug_levels)
    n2=len(new_drug_levels)
    degree_of_freedom= n1+n2-2
    alpha= 0.05
    if p_value > alpha:
        print("fail to Reject the null_hypothesis")
    else:
        print("Reject the null_hypothesis")
    print("The value of t_statistics is :",t_statistics)
    print("The value of P_value is :",p_value)
analysis_clinical_trial(existing_drug_levels,new_drug_levels)
```

```
Reject the null_hypothesis
The value of t_statistics is : 4.14048098620866
The value of P_value is : 0.0006143398442372505
```

In [42]:

```
# Q17.A school district introduces an educational intervention program to improve math scores. Write a Python
# function to analyze pre- and post-intervention test scores, calculating the t-statistic and p-value to
# determine if the intervention had a significant impact.
pre_intervention_scores = [80, 85, 90, 75, 88, 82, 92, 78, 85, 87]
post_intervention_scores = [90, 92, 88, 92, 95, 91, 96, 93, 89, 93]

def analysis_pre_post_intervention(pre_intervention_scores,post_intervention_scores):
    mean_pre= sum(pre_intervention_scores)/len(pre_intervention_scores)
    mean_post = sum(post_intervention_scores)/len(post_intervention_scores)

    null_hypothesis="The mean math scores before the intervention are equal to the mean math scores after the in
    alternate_hypothesis = "The mean math scores before the intervention are not equal to the mean math scores a
    t_statistics, p_value = stats.ttest_ind(pre_intervention_scores,post_intervention_scores)


    n1=len(pre_intervention_scores)
    n2= len(post_intervention_scores)
    degree_of_freedom= n1+n2-2
    alpha=0.05
```

```
    if p_value >alpha :
        print("Fail to reject null_hypothesis")
    else:
        print("Reject the null_hypothesis")


    print("The value of t_statistics is :",t_statistics)
    print("The value of P_value is :",p_value)
analysis_pre_post_intervention(pre_intervention_scores,post_intervention_scores)
```

```
Reject the null_hypothesis
The value of t_statistics is : -4.080355128162116
The value of P_value is : 0.0007022570725706455
```

In [43]:

```
# Q18. An HR department wants to investigate if there's a gender-based salary gap within the company. Develop
# a program to analyze salary data, calculate the t-statistic, and determine if there's a statistically
# significant difference between the average salaries of male and female employees.
# Generate synthetic salary data for male and female employees

np.random.seed(0)
male_salaries = np.random.normal(loc=50000, scale=10000, size=20)
female_salaries = np.random.normal(loc=55000, scale=9000, size=20)
def analysis_salary_data(male_salaries,female_salaries):
    mean_male_sal= sum(male_salaries)/len(male_salaries)
    mean_female_sal = sum(female_salaries)/len(female_salaries)
    null_hypothesis="The mean salary of male employees is equal to the mean salary of female employees."
    alternate_hypothesis = "The mean salary of male employees is not equal to the mean salary of female employe
    t_statistics,p_value = stats.ttest_ind(male_salaries,female_salaries)
    alpha=0.05


    if p_value >alpha :
        print("Fail to reject null_hypothesis")
    else:
        print("Reject the null_hypothesis")
    print("The t_statistics value is :",t_statistics)
    print("The value of p_value is :",p_value)
analysis_salary_data(male_salaries,female_salaries)
```

```
Fail to reject null_hypothesis
The t_statistics value is : 0.06114208969631383
The value of p_value is : 0.9515665020676465
```

In [44]:

```
# Q19. A manufacturer produces two different versions of a product and wants to compare their quality scores.
# Create a Python function to analyze quality assessment data, calculate the t-statistic, and decide
# whether there's a significant difference in quality between the two versions.
version1_scores = [85, 88, 82, 89, 87, 84, 90, 88, 85, 86, 91, 83, 87, 84, 89, 86, 84, 88, 85, 86, 89, 90, 87, 
version2_scores = [80, 78, 83, 81, 79, 82, 76, 80, 78, 81, 77, 82, 80, 79, 82, 79, 80, 81, 79, 82, 79, 78, 80, 
def analysis_quality_assessment(version1_scores,version2_scores):
    mean_vers1_score= sum(version1_scores)/len(version1_scores)
    mean_vers2_score = sum(version2_scores)/len(version2_scores)
    null_hypothesis = "The mean quality score of version 1 is equal to the mean quality score of version 2.(mea
    alternate_hypothesis = "The mean quality score of version 1 is not equal to the mean quality score of versi
    t_statistics,p_value = stats.ttest_ind(version1_scores,version2_scores)
    alpha = 0.05
    if p_value > alpha:
        print("Fail to reject null_hypothesis")
    else:
        print("Reject the null_hypothesis")
    print("The t_statistics value is :",t_statistics)
    print("The value of p_value is :",p_value)
analysis_quality_assessment(version1_scores,version2_scores)
```

```
Reject the null_hypothesis
The t_statistics value is : 11.325830417646698
The value of p_value is : 3.6824250702873965e-15
```

In [45]:

```
# Q20. A restaurant chain collects customer satisfaction scores for two different branches. Write a program to
# analyze the scores, calculate the t-statistic, and determine if there's a statistically significant differenc
# customer satisfaction between the branches.
branch_a_scores = [4, 5, 3, 4, 5, 4, 5, 3, 4, 4, 5, 4, 4, 3, 4, 5, 5, 4, 3, 4, 5, 4, 3, 5, 4, 4, 5, 3, 4, 5, 4]
branch_b_scores = [3, 4, 2, 3, 4, 3, 4, 2, 3, 3, 4, 3, 3, 2, 3, 4, 4, 3, 2, 3, 4, 3, 2, 4, 3, 3, 4, 2, 3, 4, 3]
def analysis_satis_score(branch_a_scores,branch_b_scores):
    mean_branch_a= sum(branch_a_scores)/len(branch_a_scores)
    mean_branch_b= sum(branch_b_scores)/len(branch_b_scores)
    null_hypothesis = "The mean customer satisfaction score for branch 1 is equal to the mean customer satisfac
    alternate_hypothesis = "The mean customer satisfaction score for branch 1 is not  equal to the mean custome
    t_statistics,p_value = stats.ttest_ind(branch_a_scores,branch_b_scores)
    alpha = 0.05
    if p_value >alpha :
        print("fail to reject the null hypothesis")
    else:
        print("Reject null hypothesis.")
```

```
    print("The t_statistics value is :",t_statistics)
    print("The value of p_value is :",p_value)
analysis_satis_score(branch_a_scores,branch_b_scores)
```

```
Reject null hypothesis.
The t_statistics value is : 5.480077554195743
The value of p_value is : 8.895290509945657e-07
```

In [46]:

```
# Q21.A political analyst wants to determine if there is a significant association between age groups and voter
# preferences (Candidate A or Candidate B). They collect data from a sample of 500 voters and classify
# them into different age groups and candidate preferences. Perform a Chi-Square test to determine if
# there is a significant association between age groups and voter preferences.
np.random.seed(0)
age_groups = np.random.choice(['18-30', '31-50', '51+'], size=30)
voter_preferences = np.random.choice(['Candidate A', 'Candidate B'], size=30)
df = pd.DataFrame({'Age_groups': age_groups, 'Voter_preferences': voter_preferences})
table = pd.crosstab(df['Age_groups'], df['Voter_preferences'])

observed_values = table.values
stat_test, p_value, deg_free, expected_val = stats.chi2_contingency(observed_values)

chisquare_test = sum(((o - e) ** 2) / e for o, e in zip(observed_values.flatten(), expected_val.flatten()))
print("Chi-Square Statistic (manual):", chisquare_test)
print("p-value:", p_value)
print("Degrees of Freedom:", deg_free)



null_hypothesis = "age group and voter preference are independent."
alternate_hypothesis = "age group and voter preference are not independent."
alpha = 0.05

if p_value < alpha:
    print("Reject the null hypothesis")
else:
    print("Fail to reject null hypothesis")
```

```
Chi-Square Statistic (manual): 1.4401669758812612
p-value: 0.48671161971286614
Degrees of Freedom: 2
Fail to reject null hypothesis
```

In [47]:

```
# Q22. A company conducted a customer satisfaction survey to determine if there is a significant relationship
# between product satisfaction levels (Satisfied, Neutral, Dissatisfied) and the region where customers are
# located (East, West, North, South). The survey data is summarized in a contingency table. Conduct a Chi-Square
# customer regions.
#Sample data: Product satisfaction levels (rows) vs. Customer regions (columns)

from scipy.stats import chi2_contingency

data = np.array([[50, 30, 40, 20], [30, 40, 30, 50], [20, 30, 40, 30]])
product_satisfaction_level=['Satisfied','Neutral','Dissatisfied']
customer_regions=['East','West','North','South']
df=pd.DataFrame(data,index=product_satisfaction_level,columns=customer_regions)


observed_values=df.values
stats_test,p_value,degree_freedom,expected_valu=chi2_contingency(observed_values)

chisquare_test = sum(((o - e) ** 2) / e for o, e in zip(observed_values.flatten(), expected_valu.flatten()))

print("The Value Chisquare stats (scipy) :",stats_test)
print("The Value Chisquare stats (manually) :",chisquare_test)
print("The value of p_value :",p_value)
print("The degree of freedom is :",degree_freedom)

null_hypothesis= "product satisfaction levels are independent of customer regions."
alternate_hypothesis="product satisfaction levels are not independent of customer regions."


alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis")
else:
    print("Fail to reject null hypothesis")
```

```
The Value Chisquare stats (scipy) : 27.777056277056275
The Value Chisquare stats (manually) : 27.777056277056275
The value of p_value : 0.00010349448486004387
The degree of freedom is : 6
Reject the null hypothesis
```

In [48]:

```
# Q23. A company implemented an employee training program to improve job performance (Effective, Neutral,
# Ineffective). After the training, they collected data from a sample of employees and classified them based
```

```
# on their job performance before and after the training. Perform a Chi-Square test to determine if there is a
# significant difference between job performance levels before and after the training.

from scipy.stats import chi2_contingency

data = np.array([[50, 30, 20], [30, 40, 30], [20, 30, 40]])
job_performances_before=['Effective_B','Neutral_B','Ineffective_A']
job_performances_after=['Effective_A','Neutral_A','Ineffective_A']
df= pd.DataFrame(data,index=job_performances_before,columns=job_performances_after)

observed_values= df.values
stats_test,p_value,degree_freedom,expected_value=chi2_contingency(observed_values)

chisquare_test = sum(((o - e) ** 2) / e for o, e in zip(observed_values.flatten(), expected_value.flatten()))
print("The Value of chisquare value (scipy):",stats_test)
print("The value of chisquare value (manually):",chisquare_test)
print("The value of P-value is :",p_value)
print("The degree of freedom :",degree_freedom)


null_hypothesis=" the distribution of job performance levels before training is the same as the distribution of
alternate_hypothesis = "There is a significant difference between job performance levels before and after the t
alpha= 0.05

if p_value < alpha :
    print("Reject the null hypothesis")
else:
    print("Fail the reject the null hypothesis")
```

```
The Value of chisquare value (scipy): 22.161728395061726
The value of chisquare value (manually): 22.16172839506173
The value of P-value is : 0.00018609719479882554
The degree of freedom : 4
Reject the null hypothesis
```

In [49]:
```
# Q24. A company produces three different versions of a product: Standard, Premium, and Deluxe. The
# company wants to determine if there is a significant difference in customer satisfaction scores among the
# three product versions. They conducted a survey and collected customer satisfaction scores for each
# version from a random sample of customers. Perform an ANOVA test to determine if there is a significant
# difference in customer satisfaction scores.
# Sample data: Customer satisfaction scores for each product version

standard_scores = [80, 85, 90, 78, 88, 82, 92, 78, 85, 87]
premium_scores = [90, 92, 88, 92, 95, 91, 96, 93, 89, 93]
deluxe_scores = [95, 98, 92, 97, 96, 94, 98, 97, 92, 99]

null_hypothesis ="The mean of all the group_score is same"
alternate_hypothesis = "mean is not same for atleast one group"


df = pd.DataFrame({'Standard_Score':standard_scores,'Premium_Score':premium_scores,'Deluxe_Score':deluxe_scores
f_statistics,p_value = stats.f_oneway(df['Standard_Score'],df['Premium_Score'],df['Deluxe_Score'])

print("The value of F statistics is ",f_statistics)
print("The value of P_value is :",p_value)
alpha= 0.05

if p_value < alpha :
    print("Reject the null hypothesis")
else:
    print("Fail to reject the null hypothesis")
```

```
The value of F statistics is  27.03556231003039
The value of P_value is : 3.578632885734896e-07
Reject the null hypothesis
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: