INNOMATICS
RESEARCH LABS

Name: Najmush Saquib Ali

(Data Science Internship Batch OCT-21)

! Descriptive Statistics and Python Implementation !

" Descriptive statistics are brief descriptive coefficients that summarize a given data set, which can be either a representation of the entire population or a sample of a population. "

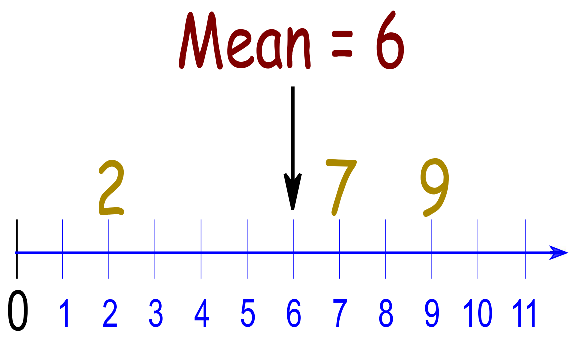
Topics:

1.Mean
2.Median
3.Mode
4.Variance
5.Standard Deviation
6.Correlation
7.Normal Distribution
8.Feature of Normal Distribution
9.Positively Skewed & Negatively Skewed Normal Distribution
10.Effect on Mean, Median and Mode due to Skewness
11.Explain QQ Plot and show the implementation of the same
12.Explain Box Cox and show the implementation of the same

The Mean

The mean is a fancy statistical way to say "average." In statistics, it is a measure of **central tendency** of a probability distribution along with **median** and **mode**.

Example:



Formulas:

Population Mean : μ

$$\mu = \sum \frac{x}{N}$$

Sample Mean : $\bar{x}$

$$\bar{x} = \sum \frac{x}{n}$$

Geometric Mean :

$$GM(x) = \sqrt[n]{x_1 \times x_2 \times \cdots \times x_n}$$

Harmonic Mean :

$$\frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \cdots + \frac{1}{x_n}}$$

Code Examples:

<u>With Library function :</u>

In [21]:
```
# It Will ignore some unneccessery warnings
import warnings
warnings.filterwarnings('ignore')
```

In [22]:
```
# Importing some neccessary modules
import pandas as pd
import seaborn as sn
df = pd.read_csv("data/data.csv")
df.head()
```

Out[22]:

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annual_HH_Incom
0	5000	8000	3	2000	6420
1	6000	7000	2	3000	7992
2	10000	4500	2	0	11280
3	10000	2000	1	0	9720
4	12500	12000	2	3000	14700

In [23]:
```
df.describe() # It shows all the statistical decriptive values
```

Out[23]:

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annual_HH_In
count	50.000000	50.000000	50.000000	50.000000	5.00000
mean	41558.000000	18818.000000	4.060000	3060.000000	4.90019
std	26097.908979	12090.216824	1.517382	6241.434948	3.20135
min	5000.000000	2000.000000	1.000000	0.000000	6.42000
25%	23550.000000	10000.000000	3.000000	0.000000	2.58750
50%	35000.000000	15500.000000	4.000000	0.000000	4.47420
75%	50375.000000	25000.000000	5.000000	3500.000000	5.94720
max	100000.000000	50000.000000	7.000000	35000.000000	1.40400

In [24]:
```
df.mean() # It gives the mean of all the integer columns
```

Out[24]:
```
Mthly_HH_Income           41558.00
Mthly_HH_Expense          18818.00
No_of_Fly_Members             4.06
Emi_or_Rent_Amt            3060.00
Annual_HH_Income         490019.04
No_of_Earning_Members         1.46
dtype: float64
```

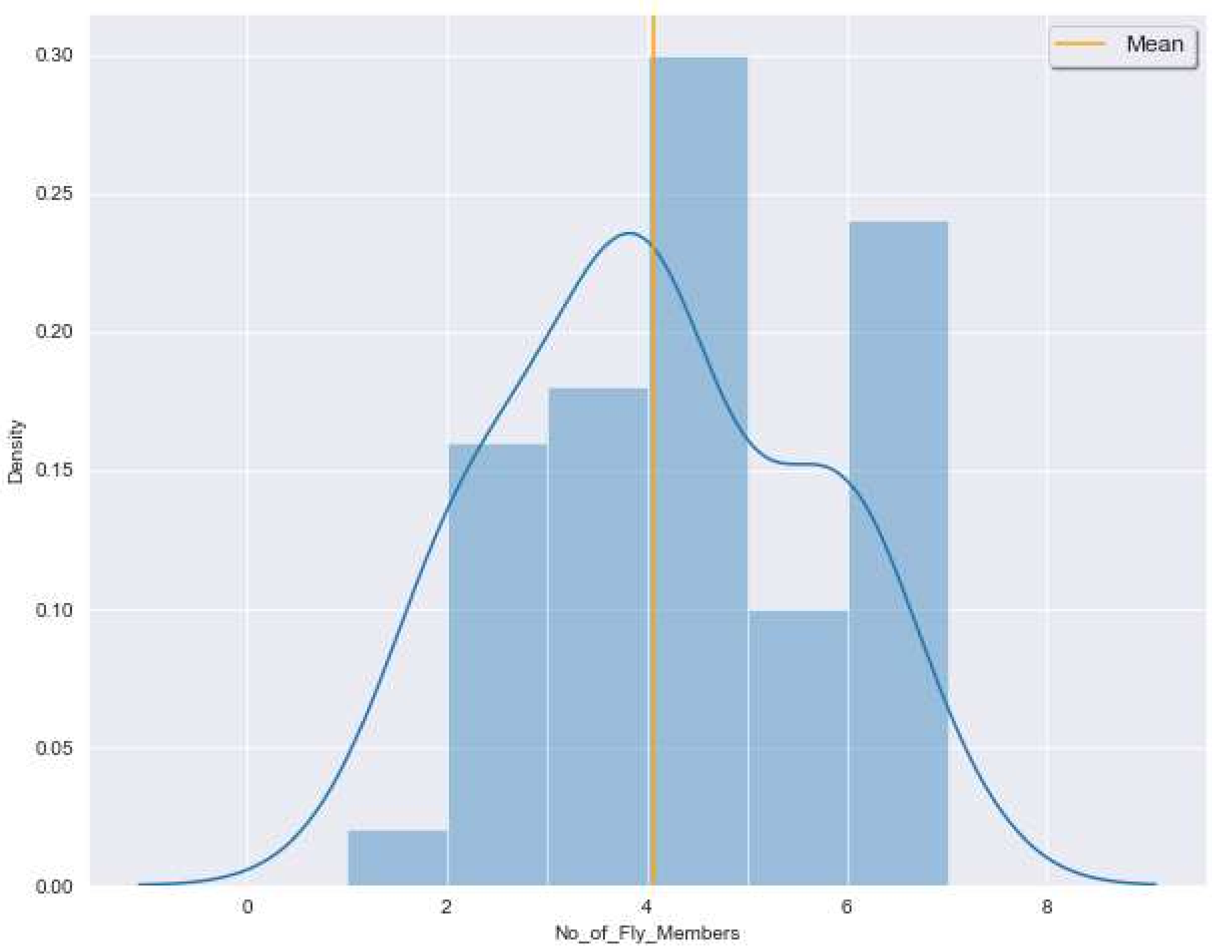
<u>Without Library function :</u>

In [25]:
```
# Taking one column from data set and finding the mean.
no_of_fly_members=list(df["No_of_Fly_Members"])
n = len(no_of_fly_members)
get_sum = sum(no_of_fly_members)
mean = get_sum / n
mean
```

Out[25]:
```
4.06
```

In [26]:
```
sn.distplot(df["No_of_Fly_Members"])
plt.axvline(df["No_of_Fly_Members"].mean() , color="orange" , label="Mean")
plt.legend(shadow=True,fontsize="larger")
```

Out[26]:
```
<matplotlib.legend.Legend at 0x24ab2824b80>
```



The Orange line is the mean of the column

The Median

The median simply gives us the middle value, as opposed to the average value of the series when ordered.In statistics, it is a measure of **central tendency** of a probability distribution.

Example:

1, 3, 3, **6**, 7, 8, 9

Median = **6**

1, 2, 3, **4**, **5**, 6, 8, 9

Median = (4 + 5) ÷ 2

= **4.5**

Formulas:

Median(if n is odd) :

$$Med = x_{\frac{(n+1)}{2}}$$

Median(if n is even) :

$$Med = \frac{x_{\frac{n}{2}} + x_{\frac{n}{(2+1)}}}{2}$$

Code Examples:

With Library function :

In [27]:
```
df.median()
```

Out[27]:
```
Mthly_HH_Income            35000.0
Mthly_HH_Expense           15500.0
No_of_Fly_Members              4.0
Emi_or_Rent_Amt                0.0
Annual_HH_Income          447420.0
No_of_Earning_Members          1.0
dtype: float64
```

Without Library function :

In [28]:
```
# Taking One Column's data to calculate the Median
Mthly_HH_Expense_lis = list(df['Mthly_HH_Expense'])
n = len(Mthly_HH_Expense_lis)
n
```

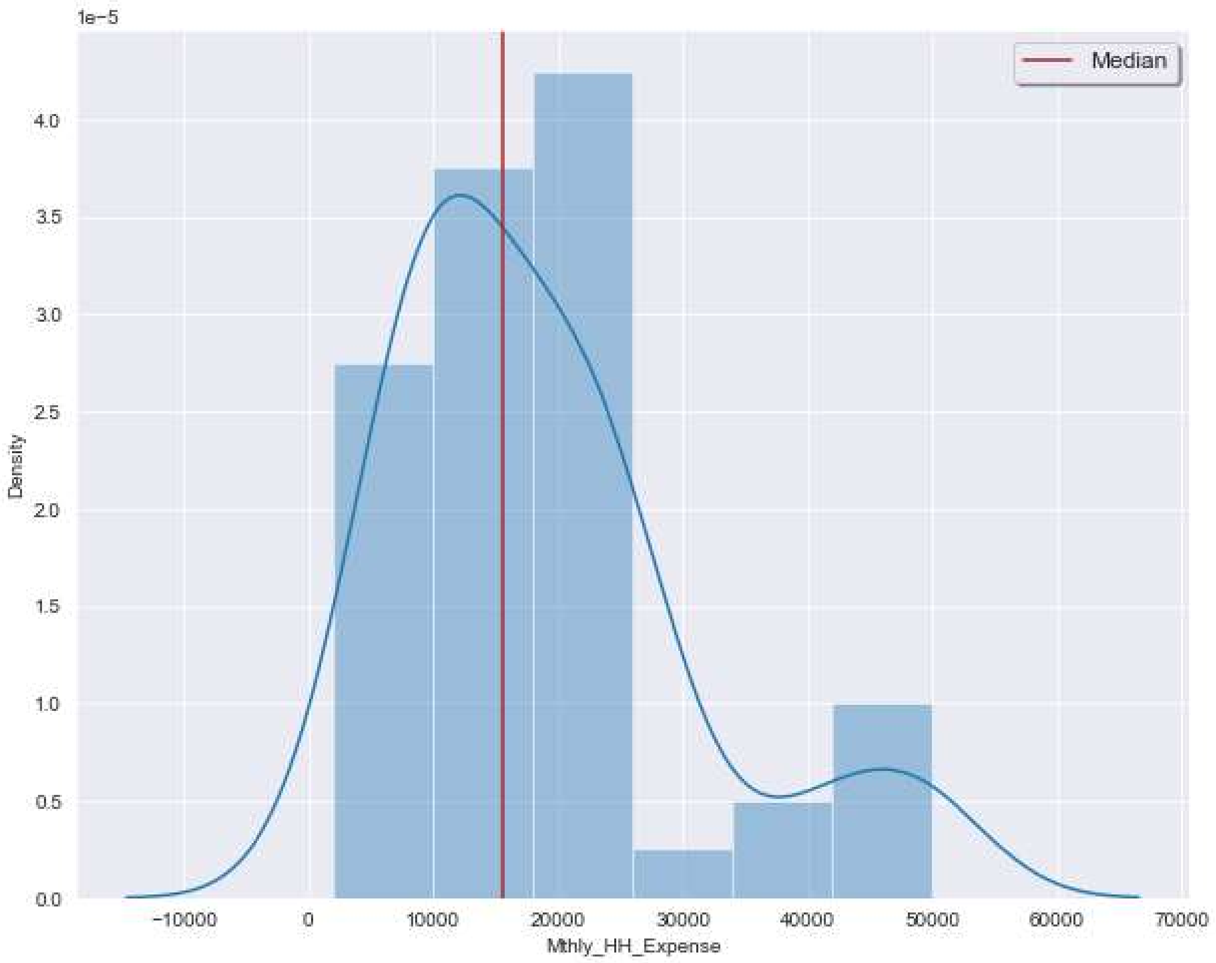
Out[28]:
```
50
```

In [29]:
```
Mthly_HH_Expense_lis.sort()

if n % 2 == 0:
    median1 = Mthly_HH_Expense_lis[n//2]
    median2 = Mthly_HH_Expense_lis[n//2 - 1]
    median = (median1 + median2)/2
else:
    median = Mthly_HH_Expense_lis[n//2]
print("Median is: " + str(median))
```

```
Median is: 15500.0
```

In [30]:
```
sn.distplot(df['Mthly_HH_Expense'])
plt.axvline(df['Mthly_HH_Expense'].median() , color="brown" , label="Median")
plt.legend(shadow=True,fontsize="larger")
```

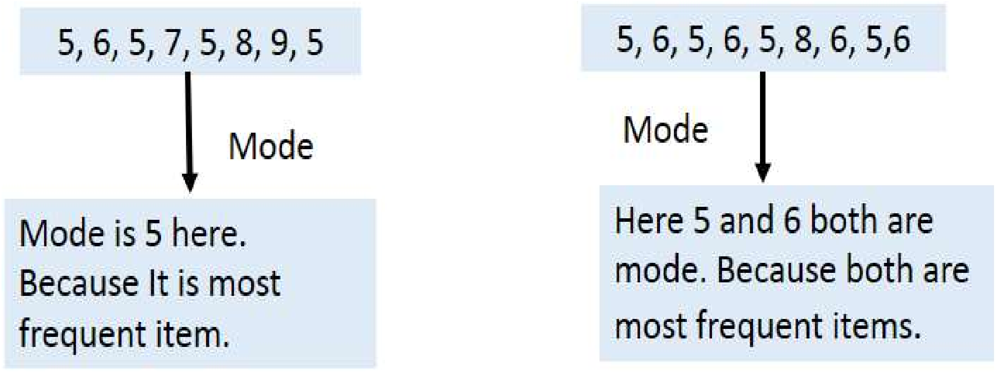
Out[30]:
```
<matplotlib.legend.Legend at 0x24ab2873cd0>
```



Note: Mean and Meadian is used for Descriptive Analysis and Data Cleaning(Filling Na Values) or Handling missing values

The Mode

Mode means most frequently occuring value in a dataset or The Observation with the highest frequency.


5, 6, 5, 7, 5, 8, 9, 5
Mode
Mode is 5 here. Because It is most frequent item.
5, 6, 5, 6, 5, 8, 6, 5,6
Mode
Here 5 and 6 both are mode. Because both are most frequent items.

Formula:

Mode :

$$L + H \frac{f_m - f_1}{(f_m - f_1) + (f_m - f_2)}$$

Code Examples:

In [31]:
```
df.columns
```

Out[31]:
```
Index(['Mthly_HH_Income', 'Mthly_HH_Expense', 'No_of_Fly_Members',
       'Emi_or_Rent_Amt', 'Annual_HH_Income', 'Highest_Qualified_Member',
       'No_of_Earning_Members'],
      dtype='object')
```

In [32]:
```
# Taking a column to calculate the mode
Highest_Q = list(df['Highest_Qualified_Member'])
```

In [33]:
```
#Using our familiar counter
from collections import Counter
Q_count = Counter(Highest_Q)
Q_count
```

Out[33]:
```
Counter({'Under-Graduate': 10,
         'Illiterate': 5,
         'Graduate': 19,
         'Post-Graduate': 6,
         'Professional': 10})
```

*****As we can see from above result that 'Graduate' is 19 so 'Graduate' is the mode here**

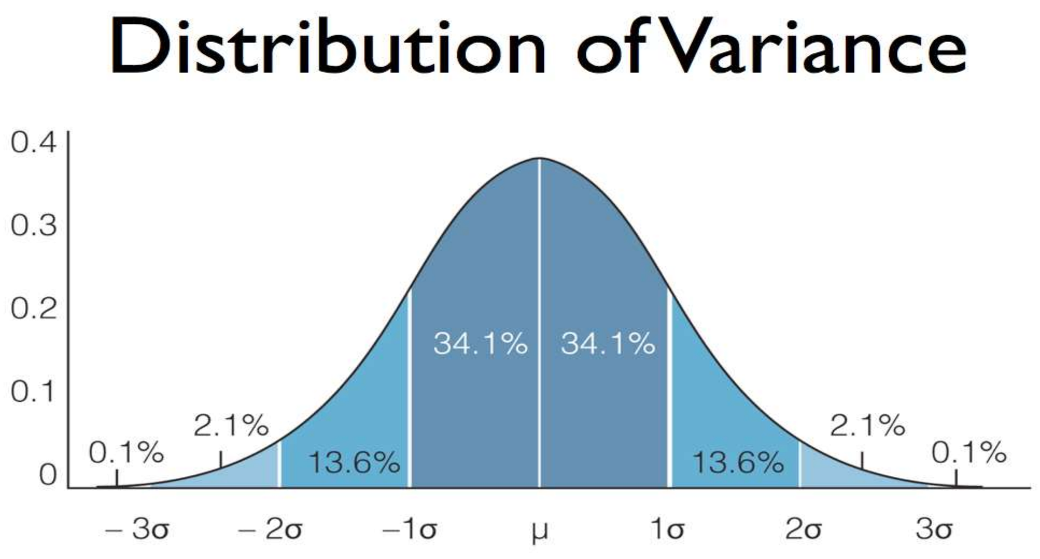
In [34]:
```
# Let's Verify
def mode(counter):
    # store a list of name:count tuples in case multiple modes
    modes = [('',0)]
    for k,v in counter.items():
        highest_count = modes[0][1]
        if v > highest_count:
            modes = [(k,v)]
        elif v == highest_count:
            modes.append((k,v))

    return modes
mode(Q_count)
```

Out[34]:
```
[('Graduate', 19)]
```

Note: Mode is not that much usefull.

The Variance

The term variance refers to a statistical measurement of the spread between numbers in a data set. More specifically, variance measures how far each number in the set is from the mean and thus from every other number in the set. Variance is often depicted by this symbol: σ2



Formulas:

Population Variation :

$$\sigma^2 = \frac{\sum (x_i - \mu)^2}{N}$$

Sample Variation :

$$\sigma^2 = \frac{\sum (x_i - \bar{x})^2}{n-1}$$

$x_i = each\, value\, from\, the\, population\, or\, sample$
$\mu = the\, population\, mean$
$N = the\, size\, of\, the\, population$
$\bar{x} = the\, sample\, mean$
$n = the\, size\, of\, the\, sample$

Note: The Sample mean is one possible position for the true population mean.

Code Examples:

With Library function :

In [35]:
```
df.std()
```

Out[35]:
```
Mthly_HH_Income           26097.908979
Mthly_HH_Expense          12090.216824
No_of_Fly_Members             1.517382
Emi_or_Rent_Amt            6241.434948
Annual_HH_Income         320135.792123
No_of_Earning_Members         0.734291
dtype: float64
```

In [36]:
```
variance_of_Mthly_HH_Income = df['Mthly_HH_Income'].std()**2
variance_of_Mthly_HH_Income
```

Out[36]:
```
681100853.0612245
```

Note: Square root of variance is standard deviation. So The Variance is the square of standard deviation.

In [37]:
```
# using numpy library
import numpy as np
np.var(list(df['Mthly_HH_Income']))
```

Out[37]:
```
667478836.0
```

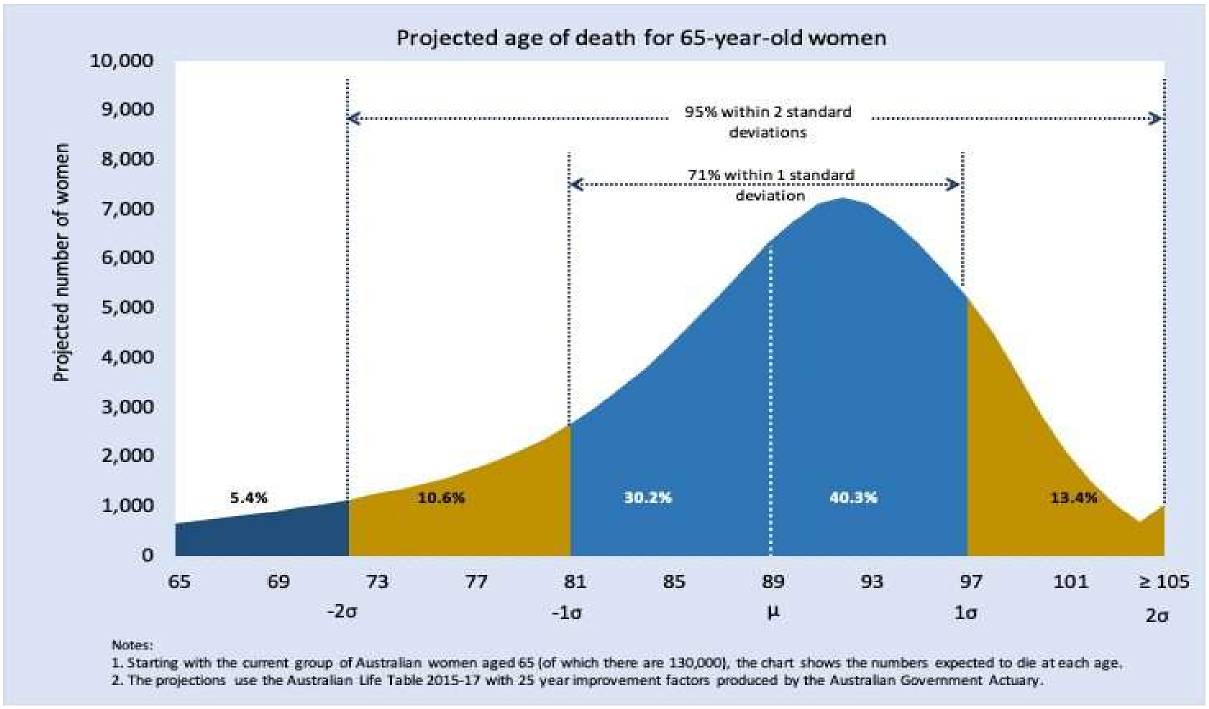
Without Library function :

In [38]:
```
lst = list(df['Mthly_HH_Income'])
avg = sum(lst) / len(lst)
var = sum((x-avg)**2 for x in lst) / len(lst)
print(var)
```

```
667478836.0
```

The Standard Deviation

Standard deviation is defined as the deviation of the data values from the average. It’s used to measure the dispersion of a data set or we can say that it is used to find how far apart our individual data points are with respect to average or how spread out data pints are.



Note: Standard Deviation is used to detect the outliers from the data set and removal of outliers using 1, 2, 3 or 4 standard deviation.Mainly we use 3 std in maximum time.

Formulas:

Population Standard Deviation :

$$\sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{N}}$$

Sample Standard Deviation :

$$\sigma = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n - 1}}$$

Code Examples:

With Library function :

In [39]:
```
df.std()
```

Out[39]:
```
Mthly_HH_Income            26097.908979
Mthly_HH_Expense           12090.216824
No_of_Fly_Members              1.517382
Emi_or_Rent_Amt             6241.434948
Annual_HH_Income          320135.792123
No_of_Earning_Members          0.734291
dtype: float64
```

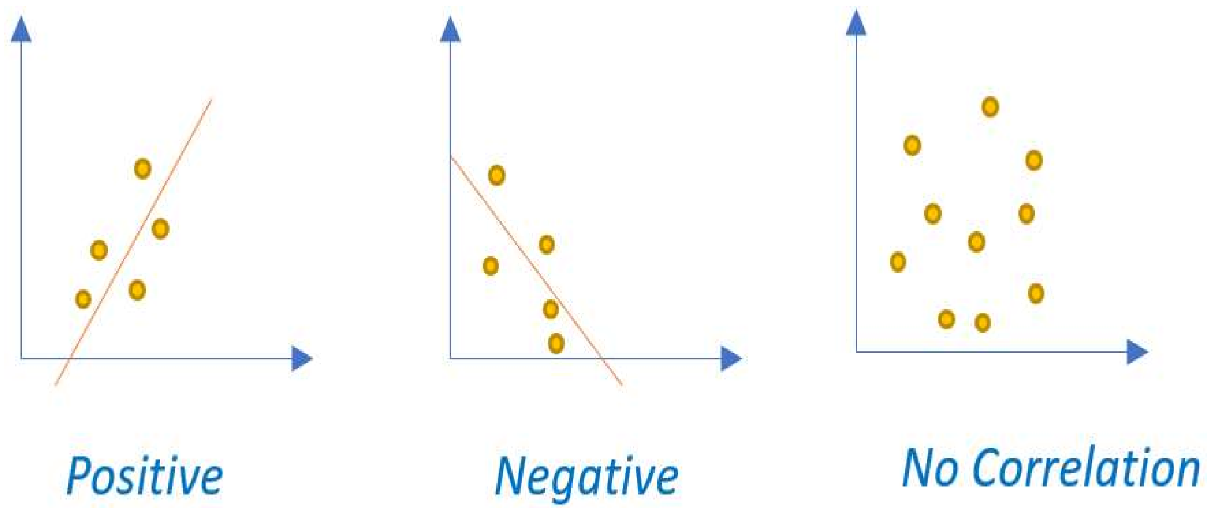
Without Library function :

In [40]:
```
# Using one column from the data set.
from math import sqrt
n= list(df['Mthly_HH_Income'])

mean =sum(n)/len(n)
SUM= 0
for i in n :
    SUM +=(i-mean)**2
stdeV = sqrt(SUM/(len(n)-1))
print(stdeV)
```

```
26097.908978713687
```

The Correlation

In statistics, correlation or dependence is any statistical relationship, whether causal or not, between two random variables or bivariate data. In the broadest sense correlation is any statistical association, though it commonly refers to the degree to which a pair of variables are linearly related. Familiar examples of dependent phenomena include the correlation between the height of parents and their offspring, and the correlation between the price of a good and the quantity the consumers are willing to purchase, as it is depicted in the so-called demand curve.



What is Pearson Correlation?

Correlation between sets of data is a measure of how well they are related. The most common measure of correlation in stats is the Pearson Correlation. The full name is the Pearson Product Moment Correlation (PPMC). It shows the linear relationship between two sets of data. In simple terms, it answers the question, Can I draw a line graph to represent the data? Two letters are used to represent the Pearson correlation: Greek letter rho (ρ) for a population and the letter “r” for a sample.

Formulas:

Pearson correlation coefficient:

$$r = \frac{\sum XY - (\sum X \sum Y)}{\sqrt{[\sum x^2 - (\sum x)^2][\sum y^2 - (\sum y)^2]}}$$

Two other formulas are commonly used: the sample correlation coefficient and the population correlation coefficient.

Sample correlation coefficient:

$$r_{xy} = \frac{s_{xy}}{s_x s_y}$$

$s_{xy}\ and\ s_y\ are\ the\ sample\ standard\ deviations,\ and\ s_{xy}\ is\ the\ sample\ covariance.$

Population correlation coefficient:

$$\rho_{xy} = \frac{\sigma_{xy}}{\sigma_x \sigma_y}$$

$The\ population\ correlation\ coefficient\ uses\ \sigma_x\ and\ \sigma_y\ as\ the\ population\ standard\ deviation$

Code Examples:

In [41]:
```
df.corr()
```

Out[41]:

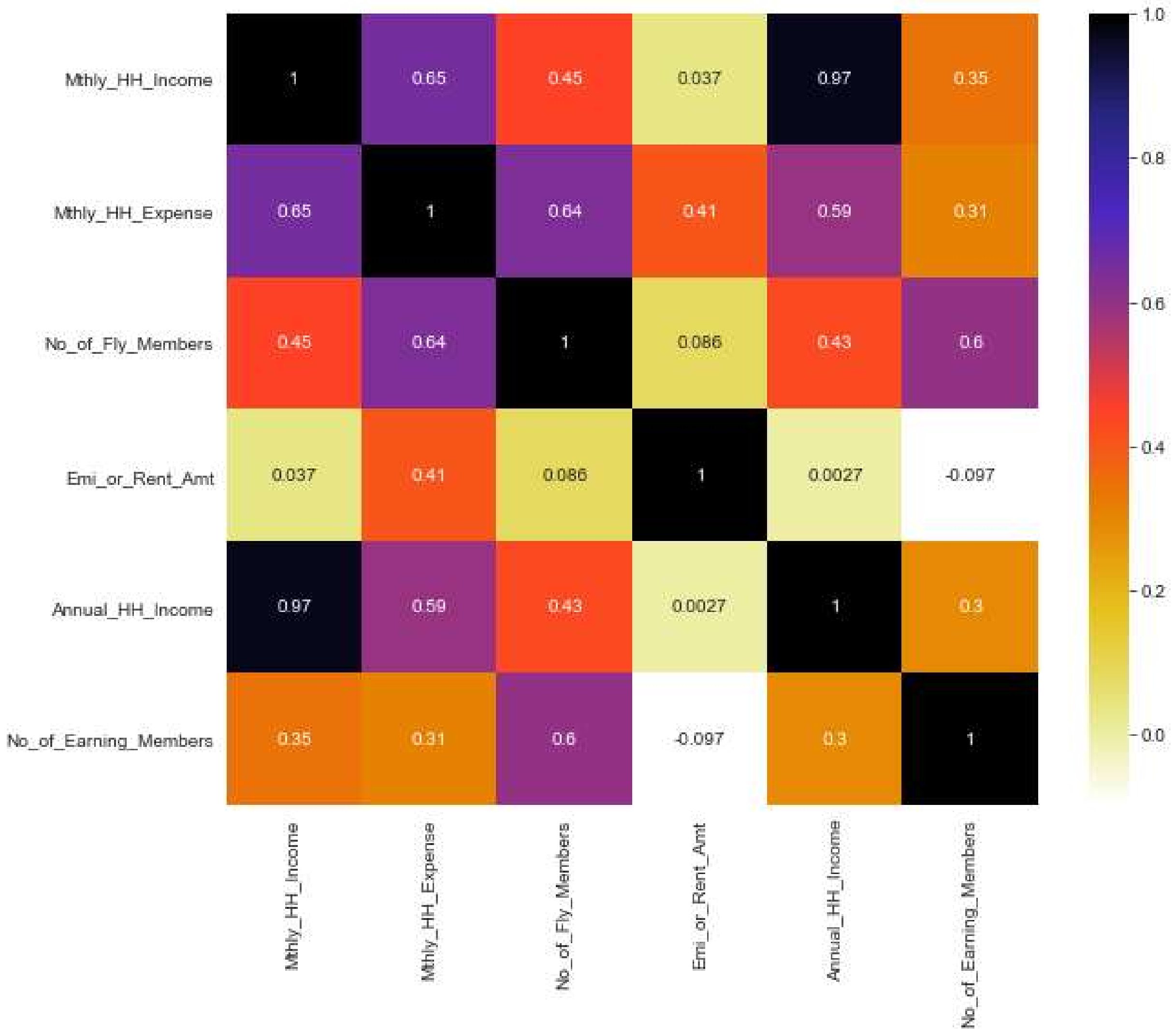
	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_/
Mthly_HH_Income	1.000000	0.649215	0.448317	0.036
Mthly_HH_Expense	0.649215	1.000000	0.639702	0.405
No_of_Fly_Members	0.448317	0.639702	1.000000	0.085
Emi_or_Rent_Amt	0.036976	0.405280	0.085808	1.000
Annual_HH_Income	0.970315	0.591222	0.430868	0.002
No_of_Earning_Members	0.347883	0.311915	0.597482	-0.097

Note: The Correlation between different columns of the data set is lie between -1 to 1 only. Negative numbers means the columns are negatively correlated and the positive numbers means the columns are positively correlated with each other.

"""By just seeing the numbers it is hard to do feature selection"""

"""Let's use Heatmap"""

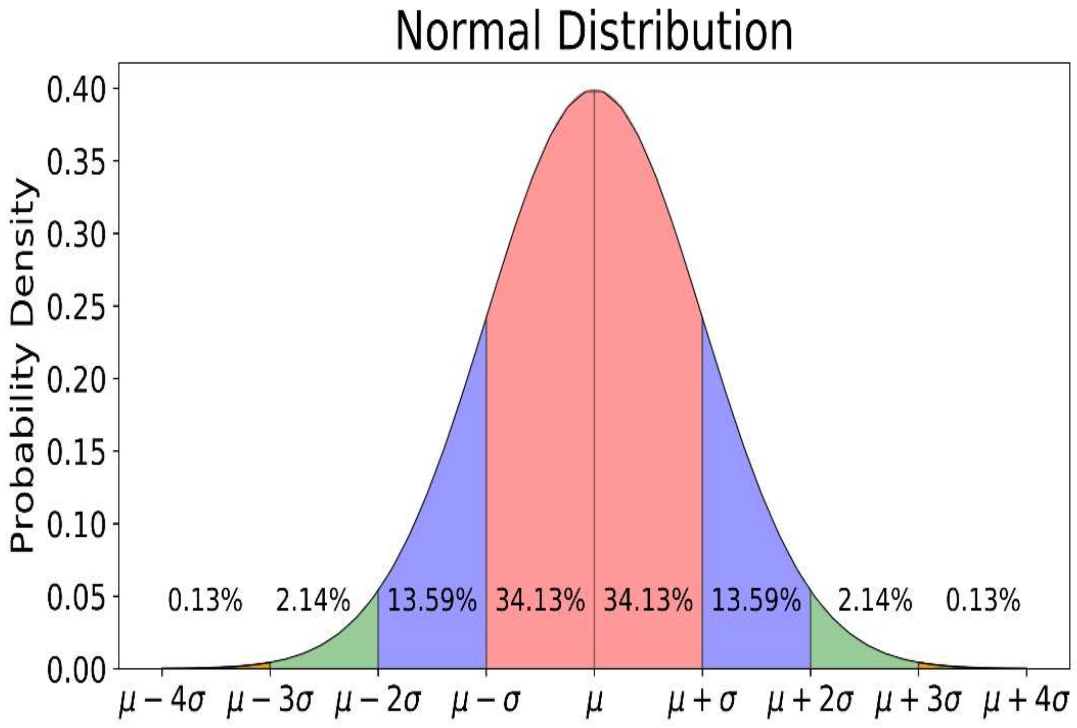
In [42]:
```
# Importing some important libraries
import matplotlib.pyplot as plt
import seaborn as sn
# Using Pearson correlation
plt.figure(figsize=(10,8))
sn.heatmap(df.corr(), annot=True, cmap=plt.cm.CMRmap_r)
plt.show()
```



	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annual_HH_Income	No_of_Earning_Members
Mthly_HH_Income	1	0.65	0.45	0.037	0.97	0.35
Mthly_HH_Expense	0.65	1	0.64	0.41	0.59	0.31
No_of_Fly_Members	0.45	0.64	1	0.086	0.43	0.6
Emi_or_Rent_Amt	0.037	0.41	0.086	1	0.0027	-0.097
Annual_HH_Income	0.97	0.59	0.43	0.0027	1	0.3
No_of_Earning_Members	0.35	0.31	0.6	-0.097	0.3	1

The Normal Distribution

Normal distribution, also known as **the Gaussian distribution**, is a probability distribution that is symmetric about the mean, showing that data near the mean are more frequent in occurrence than data far from the mean. In graph form, normal distribution will appear as a **bell curve.**The graph of the normal distribution is characterized by two parameters: the mean, or average, which is the maximum of the graph and about which the graph is always symmetric; and the standard deviation, which determines the amount of dispersion away from the mean. A small standard deviation (compared with the mean) produces a steep graph, whereas a large standard deviation (again compared with the mean) produces a flat graph.



For a normal distribution, 68% of the observations are within +/- one standard deviation of the mean, 95% are within +/- two standard deviations, and 99.7% are within +- three standard deviations.

Formula:

The normal distribution is produced by the normal density function,

$$p(x) = \frac{e^{\frac{-(x-\mu)^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}$$

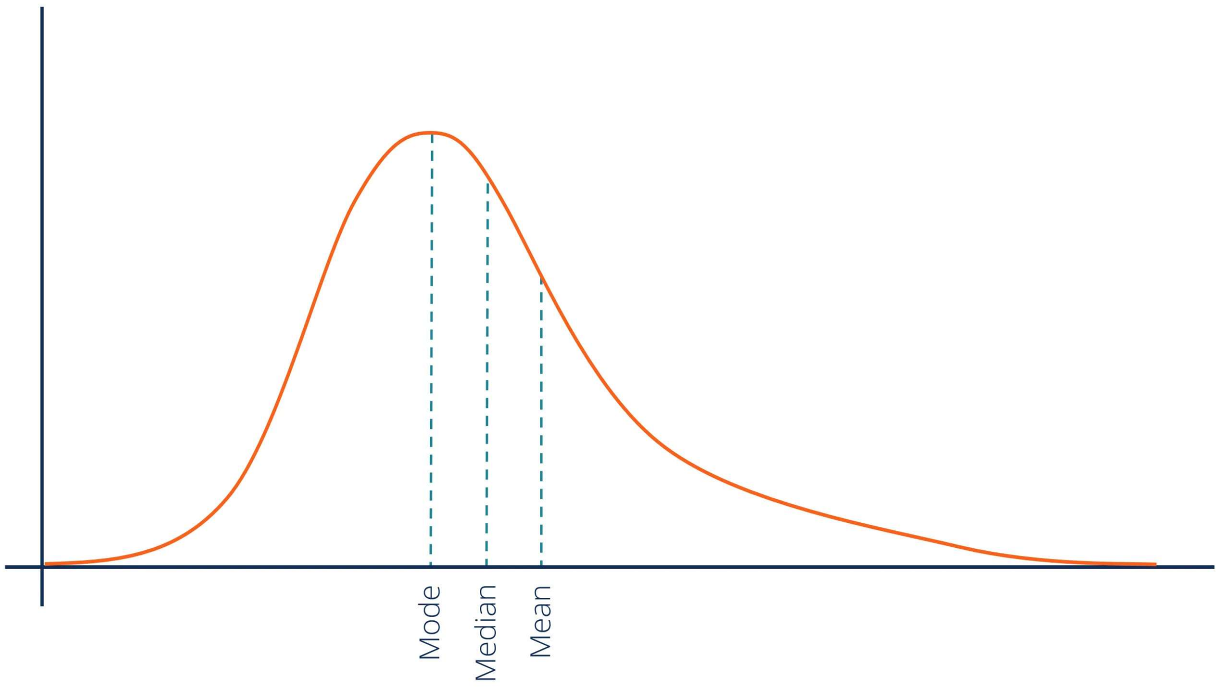
$In this exponential function e is the constant 2.71828 \ldots, \mu is the me$

Some Features of A Normal Distribution:

1.The **mean, median and mode** are exactly the same means that The Data Is Centered About the Mean-Mode-Median
2.The distribution is **symmetric** about the mean—half the values fall below the mean and half above the mean.
3.The distribution can be described by two values: **the mean and the standard deviation.**

Positively Skewed Normal Distribution:

In statistics, **a positively skewed (or right-skewed) distribution** is a type of distribution in which most values are clustered around the left tail of the distribution while the right tail of the distribution is longer. The positively skewed distribution is a direct opposite of the negatively skewed distribution.

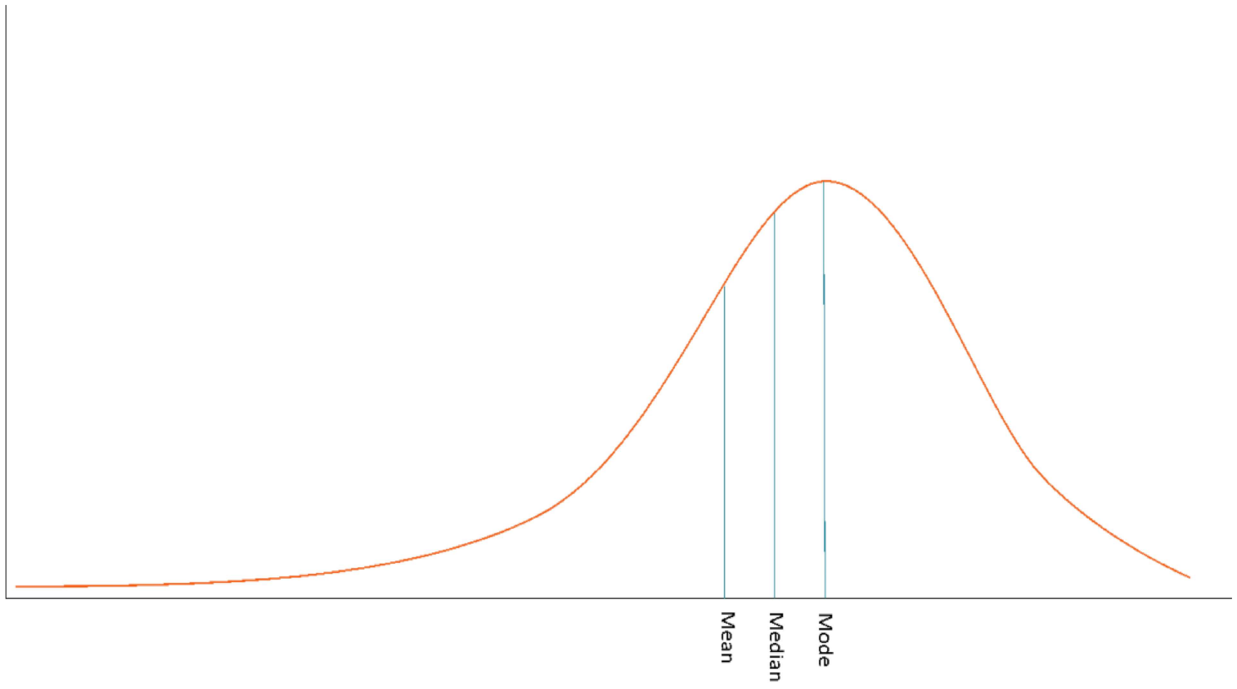


Effect on Mean, Median and Mode due to Positive Skewness:

Mean> Median> Mode

Negatively Skewed Normal Distribution:

In statistics, **a negatively skewed (also known as left-skewed) distribution** is a type of distribution in which more values are concentrated on the right side (tail) of the distribution graph while the left tail of the distribution graph is longer.



Effect on Mean, Median and Mode due to Positive Skewness:**Mean <Median <Mode**

Formula:

1.Mode Skewness:

$$skew = \frac{mean - mode}{std\,dev.}$$

2.Median Skewness:

$$skew = \frac{3(mean - median)}{std\,dev.}$$

Note:
1. Mode suckes the small data set
2. The greater the skew, the greater the distance between Mode, Median and Mean
3. We can use Normal Distribution to detect and remove the outliers.

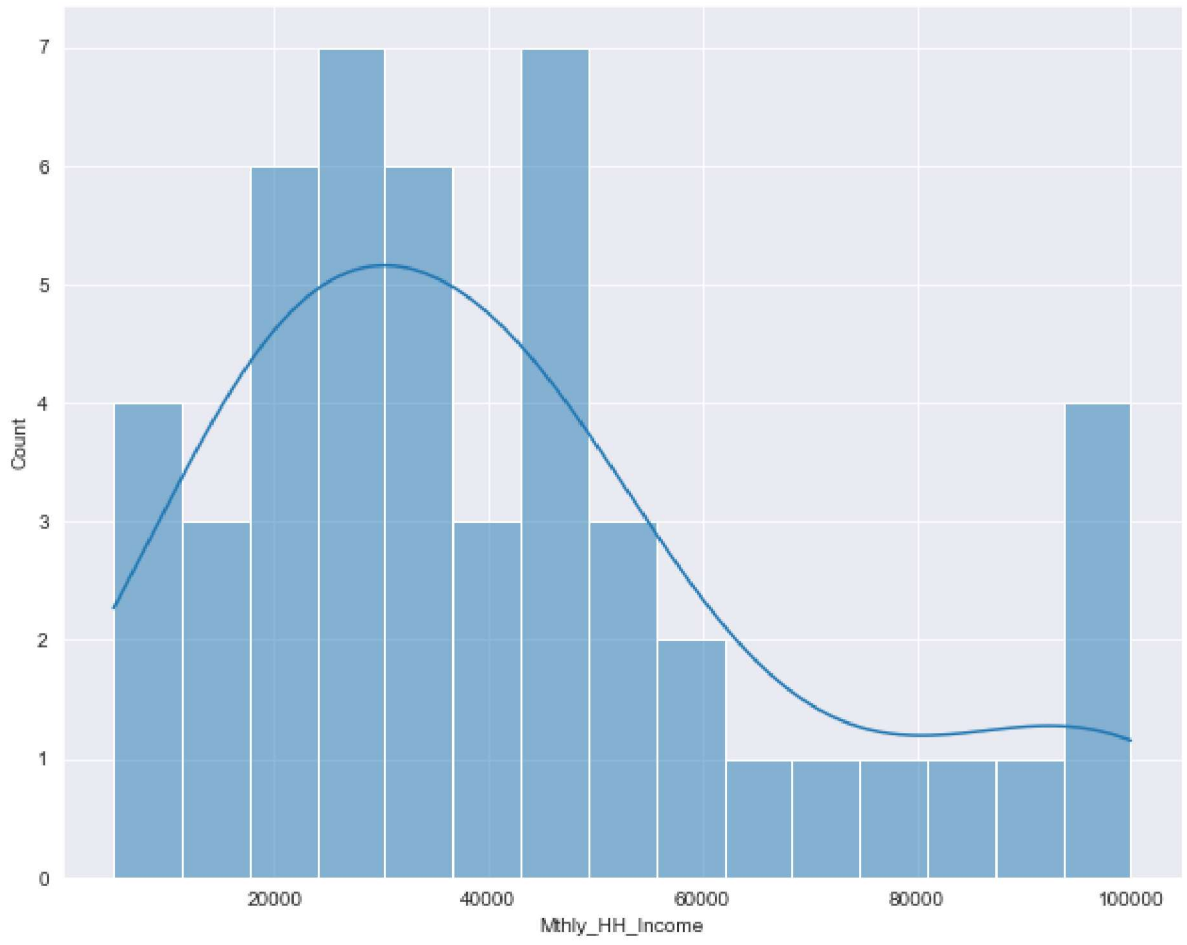
Code Examples:

In [43]:
```
df.describe()
```

Out[43]:

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annual_HH_In
count	50.000000	50.000000	50.000000	50.000000	5.00000
mean	41558.000000	18818.000000	4.060000	3060.000000	4.90019
std	26097.908979	12090.216824	1.517382	6241.434948	3.20135
min	5000.000000	2000.000000	1.000000	0.000000	6.42000
25%	23550.000000	10000.000000	3.000000	0.000000	2.58750
50%	35000.000000	15500.000000	4.000000	0.000000	4.47420
75%	50375.000000	25000.000000	5.000000	3500.000000	5.94720
max	100000.000000	50000.000000	7.000000	35000.000000	1.40400

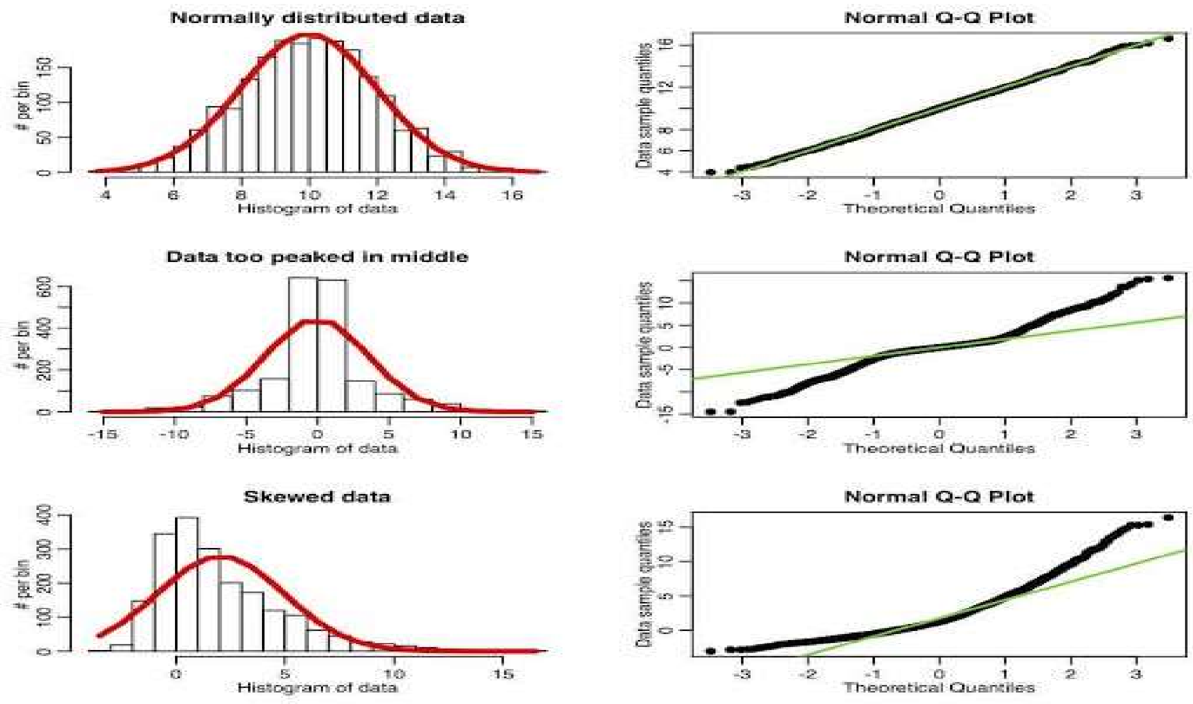
In [44]:
```
%matplotlib inline
plt.rcParams['figure.figsize'] = (10,8)
sn.histplot(df['Mthly_HH_Income'],bins=15,kde=True)
plt.show()
```



"""In the above histogram as you can see the tail is pointing towards right means it is positively skewed"""".

Q-Q(quantile-quantile) Plot

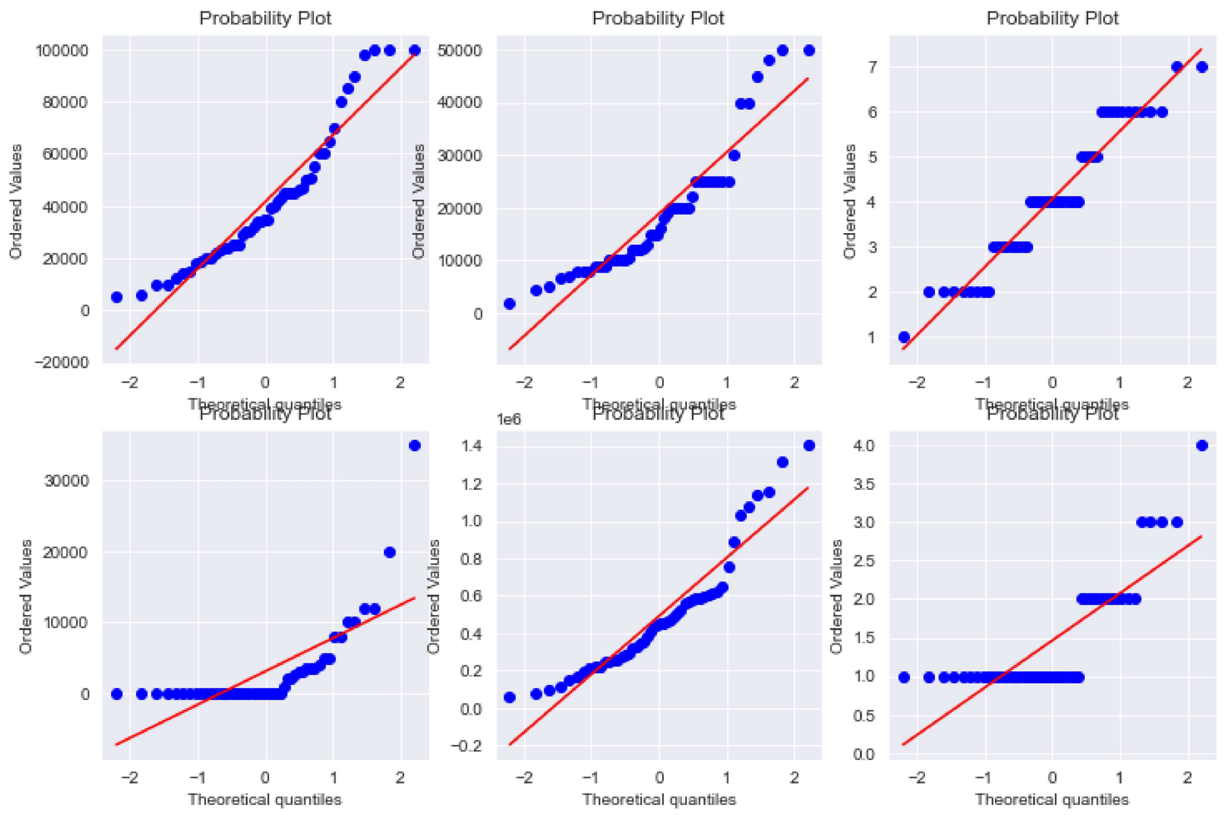
In Statistics, **Q-Q(quantile-quantile) plots** play a very vital role to graphically analyze and compare two probability distributions by plotting their quantiles against each other. If the two distributions which we are comparing are exactly equal then the points on the Q-Q plot will perfectly lie on a straight line y = x.Q-Q plots are used to find the type of distribution for a random variable whether it be a Gaussian Distribution, Uniform Distribution, Exponential Distribution or even Pareto Distribution, etc. **We can tell the type of distribution using the power of the Q-Q plot just by looking at the plot.**



Code Examples:

```
In [45]: from scipy import stats
```

In [46]:
```
# Initiating a figure
plt.figure(figsize=(12, 8))

# Creating a plot with 1 row and 2 cols

plt.subplot(2, 3, 1)
stats.probplot(df['Mthly_HH_Income'], dist="norm", plot=plt)

plt.subplot(2, 3, 2)
stats.probplot(df['Mthly_HH_Expense'], dist="norm", plot=plt)

plt.subplot(2, 3, 3)
stats.probplot(df['No_of_Fly_Members'], dist="norm", plot=plt)

plt.subplot(2, 3, 4)
stats.probplot(df['Emi_or_Rent_Amt'], dist="norm", plot=plt)

plt.subplot(2, 3, 5)
stats.probplot(df['Annual_HH_Income'], dist="norm", plot=plt)

plt.subplot(2, 3, 6)
stats.probplot(df['No_of_Earning_Members'], dist="norm", plot=plt)

plt.show()
```



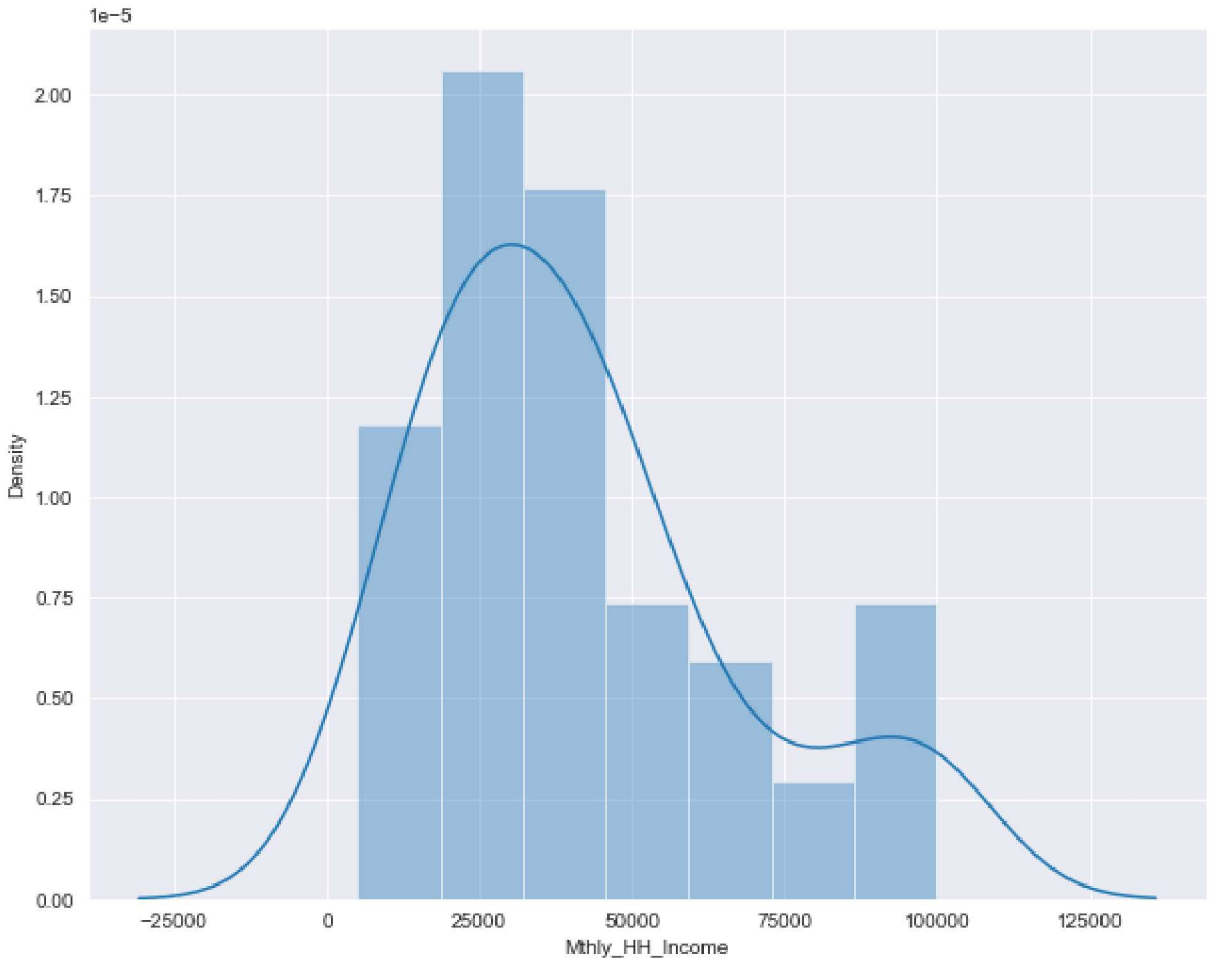
from the above plots we can get a idea that the columns are not normally distributed

Box-Cox Transformation

The Box-Cox transformation transforms our data so that it closely resembles a normal distribution.In many statistical techniques, we assume that the errors are normally distributed. This assumption allows us to construct confidence intervals and conduct hypothesis tests. By transforming your target variable, we can (hopefully) normalize our errors (if they are not already normal).Additionally, transforming our variables can improve the predictive power of our models because transformations can cut away white noise.

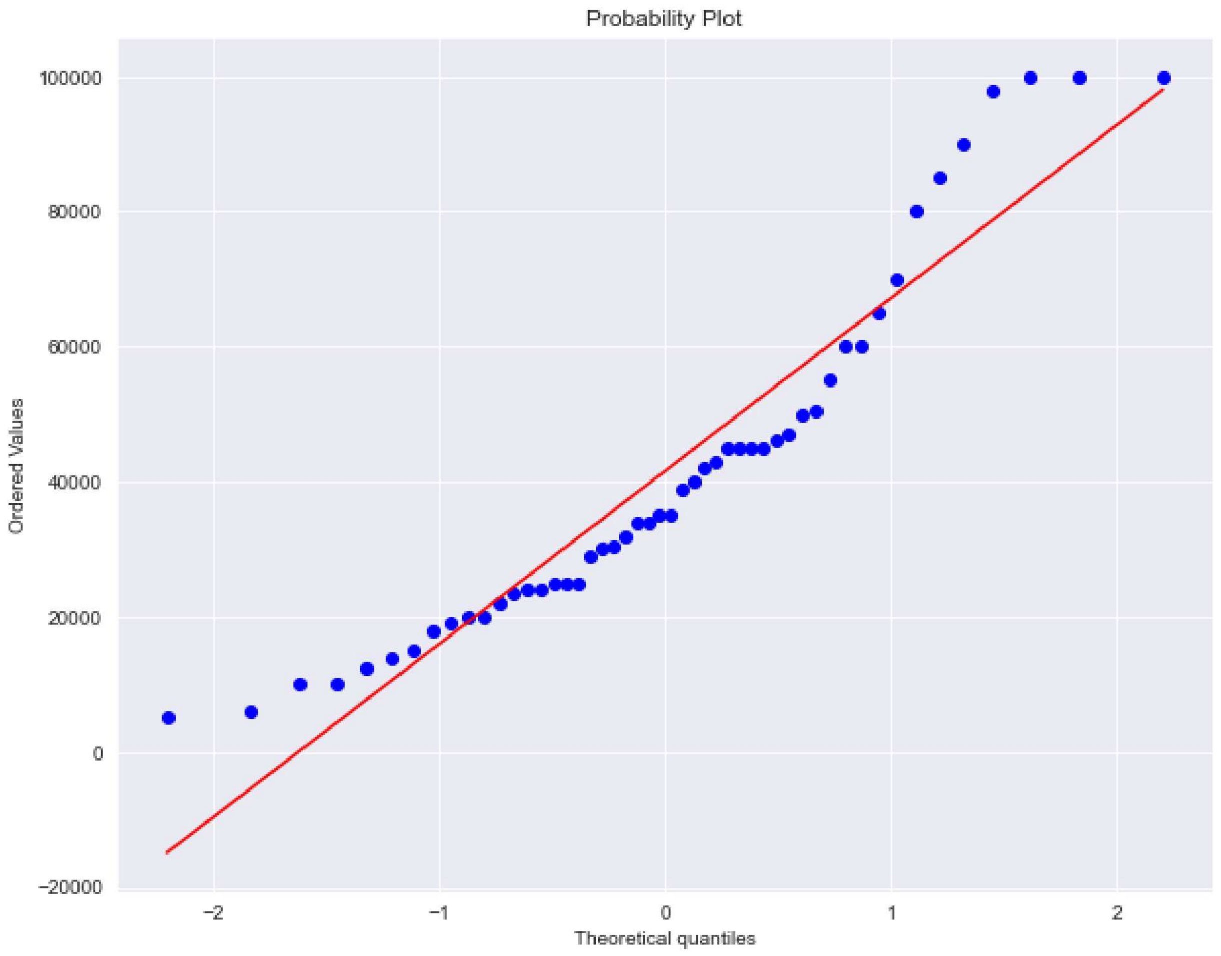
```
In [47]: sn.distplot(df['Mthly_HH_Income'])
```

```
Out[47]: <AxesSubplot:xlabel='Mthly_HH_Income', ylabel='Density'>
```



As we can see that there are some discrepancies. Let's plot a Q-Q plot to check the normality of the distribution

In [48]:
```
stats.probplot(df['Mthly_HH_Income'] , dist='norm' , plot=plt)
sn.set_style("darkgrid")
```

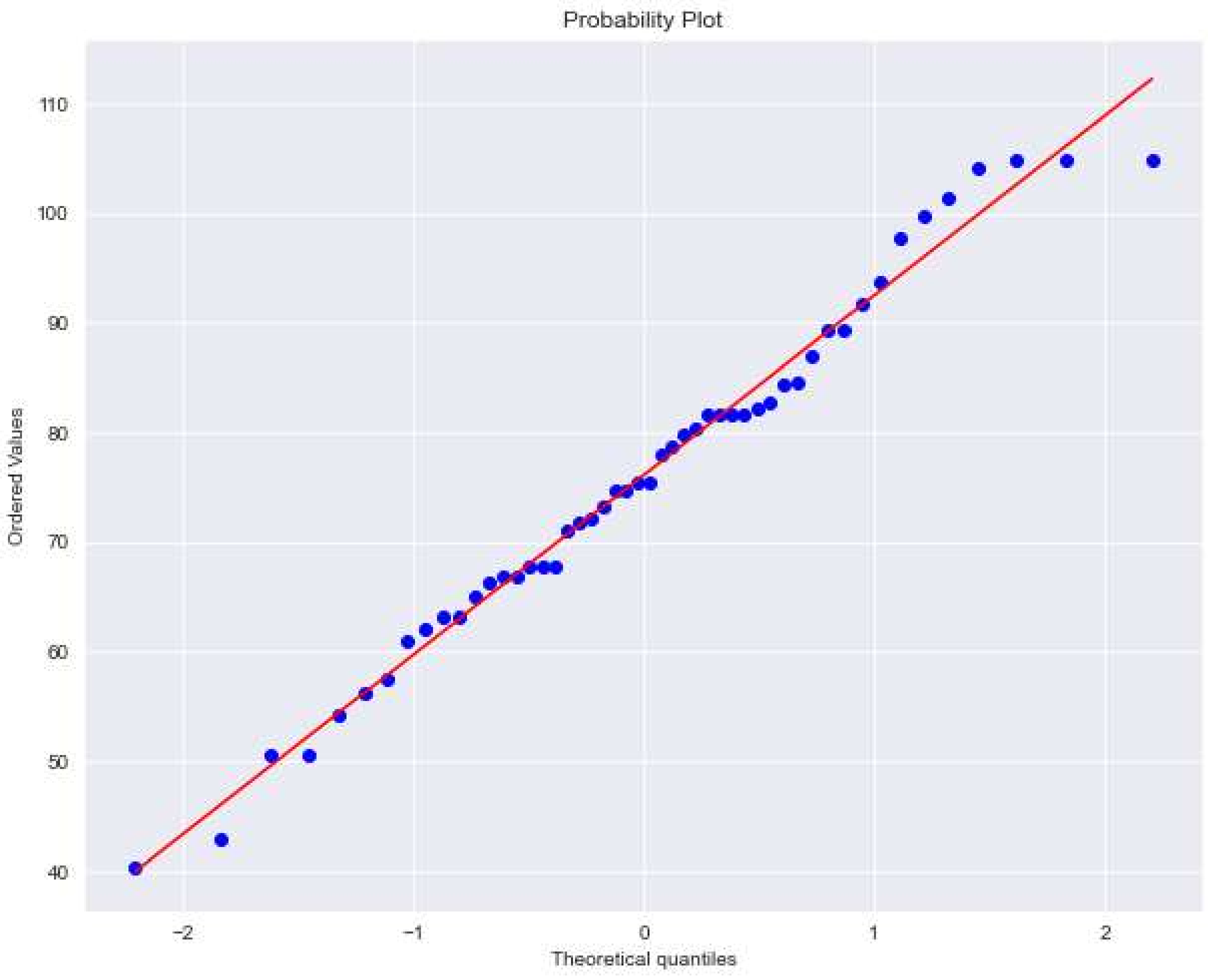


Using box-cox transformation

In [49]:
```
# pr_1 = transformed by the box-cox l=lamda
pr_1 , l = stats.boxcox(df['Mthly_HH_Income'])
print(l)
```

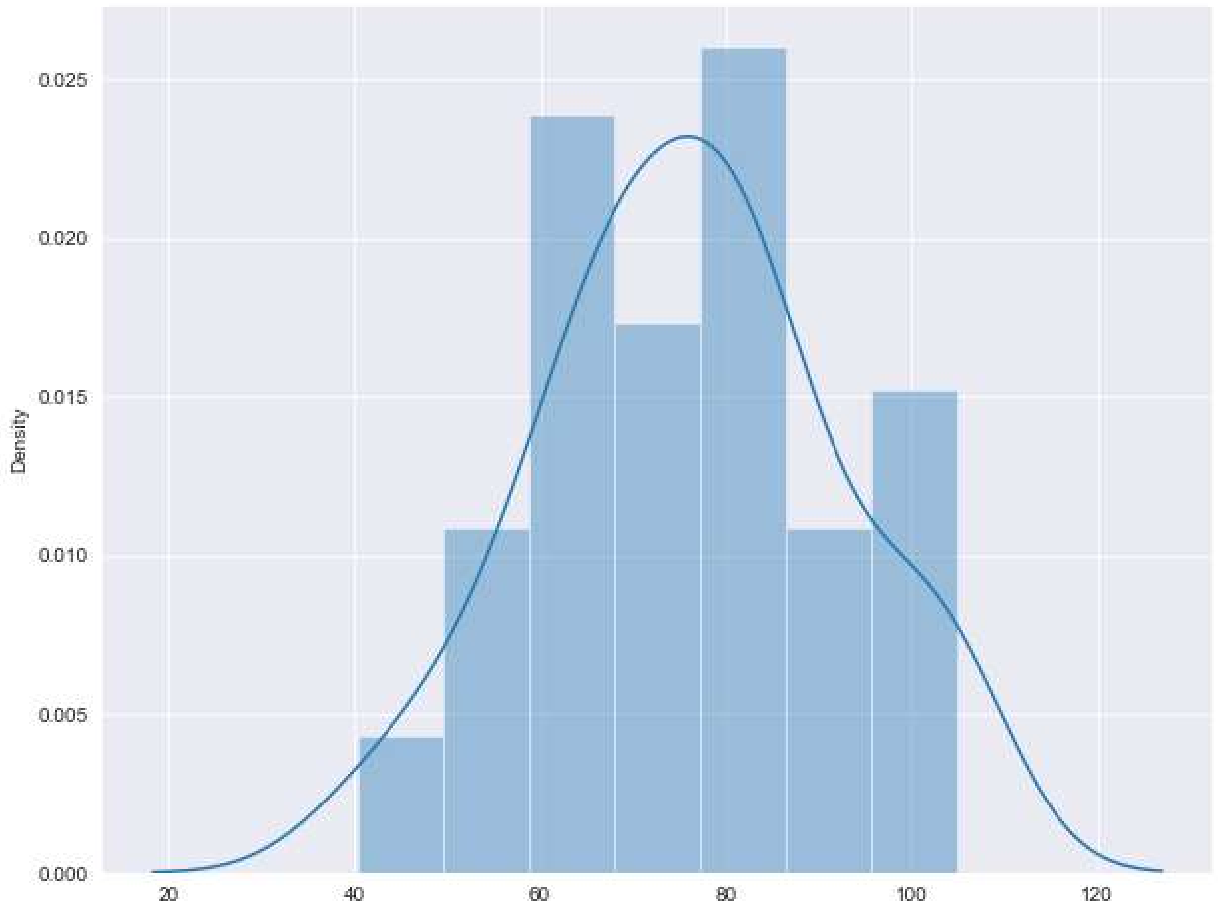
```
0.3031376789702236
```

In [50]:
```
# Ploting the Q-Q plot to see the normality of the transformed column
stats.probplot(pr_1 , dist='norm' , plot=plt)
sn.set_style("darkgrid")
```



In [51]:
```
# Ploting the dist plot to see the normality of the transformed column
sn.distplot(pr_1)
```

Out[51]:
```
<AxesSubplot:ylabel='Density'>
```



In [ ]: