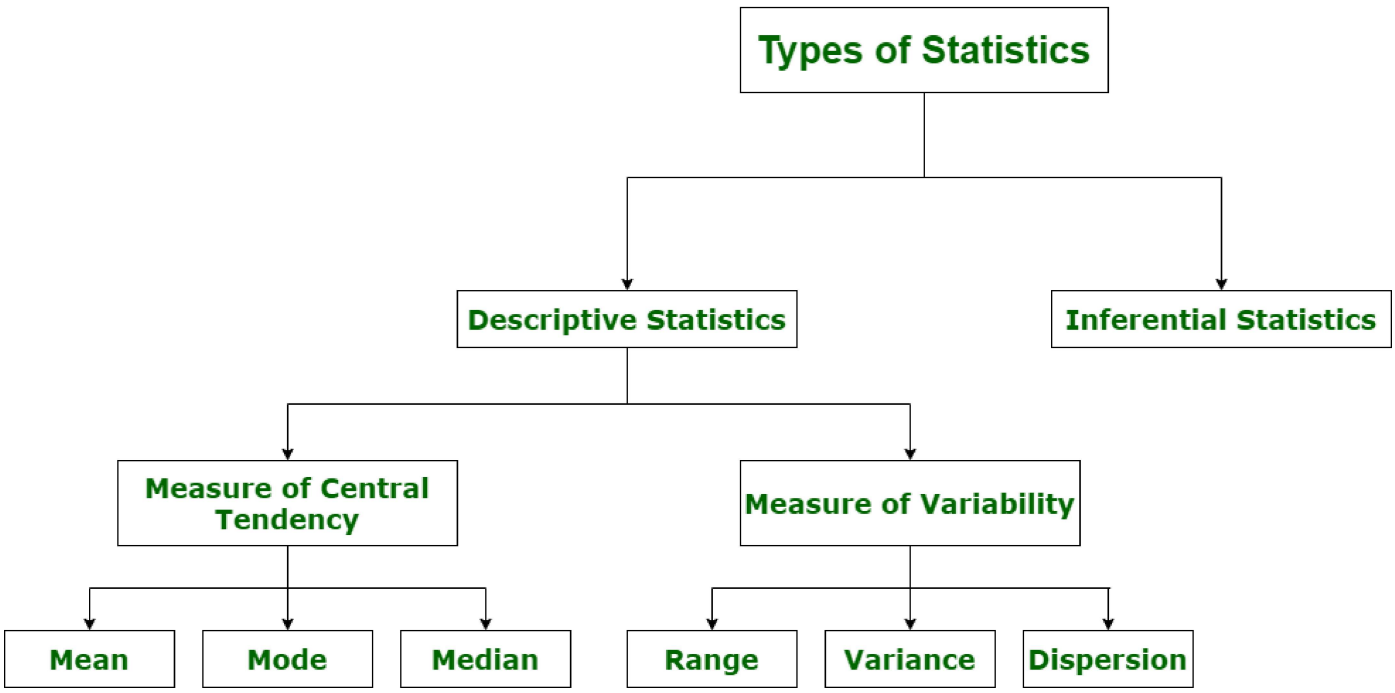
Statistics

Descriptive Statistics:

Descriptive statistics summarizes or describes the characteristics of a data set, it is an Area of applied Mathematics concern with data collections, analysis, interpretation and presentation.
These are mainly classified into two categorized into 2 typres are
1. Measures of Central Tendency
2. Measures of Spread (or Dispersion)

Measures of Central Tendency is a one number summary of the data that typically describes the center of the data. These one number summary is of three types as Mean,Median, Mode.
Measures of Dispersion describes the spread of the data around the central value (or the Measures of Central Tendency).

Classification of Statistics:



Measures of Central Tendency

Mean:

- It is defined as the ratio of the sum of all the observations in the data to the total number of observations.
- Simply Measure of all the values in sample is called as Mean.

$$represented by \ \mu$$

Formula:

$$\mu = \frac{\sum_{i=1}^{n}}{N}\mu_i$$

Where **N** denlotes no of observations

Example from Scratch and by using Libraries

In [564]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from scipy.stats import norm
from scipy.stats import skew
import pylab as p
import warnings
warnings.filterwarnings("ignore")
```

In [565]:

```
df = pd.read_csv(r"C:\Users\ameer\Desktop\innomatics\Datasets\Fmly_data.csv")
df.head(5)
```

Out[565]:

	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Rent_Amt	Annual_HH_In
0	5000	8000	3	2000	6
1	6000	7000	2	3000	7
2	10000	4500	2	0	11
3	10000	2000	1	0	9
4	12500	12000	2	3000	14

In [566]:

```
Anl_incm_sum = sum(df['Annual_HH_Income'])
Anl_incm_len = df['Annual_HH_Income'].__len__()
print(Anl_incm_sum)
print(Anl_incm_len)
```

```
24500952
50
```

In [567]:

```
Anl_incm_mean_ = (Anl_incm_sum)/(Anl_incm_len)
Anl_incm_mean_
```

Out[567]:

```
490019.04
```

In [568]:

```
Anl_incm_mean = np.mean(df['Annual_HH_Income'])
Anl_incm_mean
```

Out[568]:

```
490019.04
```

Median:

- It is the point which divides the entire data into two equal halves. One-half of the data is less than the median, and the other half is greater than the same.
- Simply Measure of central value of th sample set is called as Median
- Median is calculated by first arranging the data in either ascending or descending order.
- If the number of observations are odd, median is given by the middle observation in the sorted form.
- If the number of observations are even, median is given by the mean of the two middle observation in the sorted form.
- An important point is that the order of the data (ascending or descending) does not effect the median.

Formula:

$$median = \left\{ \frac{N+1}{2} \right\}^{th} value$$

if the observations value is **ODD**.

$$median = \left\{ \frac{\left\{ \frac{N}{2} \right\}^{th} + \left\{ \frac{N}{2} + 1 \right\}^{th}}{2} \right\} value$$

if the observations value is **EVEN**.

Example from Scratch and by using Libraries

In [569]:

```
Anl_incm = np.sort(df['Annual_HH_Income'])
Anl_incm_len = df['Annual_HH_Income'].__len__()

if Anl_incm_len % 2 == 0:
    term_1 = Anl_incm[Anl_incm_len//2]
    term_2 = Anl_incm[((Anl_incm_len//2)-1)]
    Anl_incm_median_ = (term_1 + term_2)/2
else:
    Anl_incm_median_ = Anl_incm[((Anl_incm_len-1)//2)]

Anl_incm_median_
```

Out[569]:

```
447420.0
```

In [570]:

```
Anl_incm_median = np.median(df['Annual_HH_Income'])
Anl_incm_median
```

Out[570]:

```
447420.0
```

Mode:

- Mode is the number which has the maximum frequency in the entire data set, or in other words,mode is the number that appears the maximum number of times.
- A data can have one or more than one mode.

Example from Scratch and by using Libraries

In [571]:

```
m = list(df['Annual_HH_Income'])
mode = max(m, key = m.count)
print(mode)
```

```
590400
```

In [572]:

```
import statistics as st
st.mode(df['Annual_HH_Income'])
```

Out[572]:

```
590400
```

In [573]:

```
stats.mode(df['Annual_HH_Income'])
```

Out[573]:

```
ModeResult(mode=array([590400], dtype=int64), count=array([2]))
```

Measures of Spread (or Dispersion)

- It describes how similar or varied data points for a particular variable. This broadly categorised as four types.....
- Range
- Inter Quartile Range
- Varience
- Standard Deviation

Variance:

- Variance measures how far are data points spread out from the mean.
- A high variance indicates that data points are spread widely and a small variance indicates that the data points are closer to the mean of the data set.

Formula:

$$\sigma^2 = \left\{ \frac{\Sigma_{i=1}^{n} |\mu - obj_i|}{N} \right\}$$

Where

- **N** is no of Observations,
- **obj** is individual Datapoint from a Data Set, and
- mean is

$$\mu$$

.

Example from Scratch and by using Libraries

In [574]:

```
def vrnc(x):
    mean_ = sum(x) / len(x)
    dev = [(obj_i - mean_) ** 2 for obj_i in x]
    var = sum(dev) / len(x)
    return var

x = df['Annual_HH_Income']
vrnc(x)
```

Out[574]:

```
100437186889.95831
```

In [575]:

```
Anl_incm_variance = st.pvariance(df['Annual_HH_Income'])
Anl_incm_variance
```

Out[575]:

```
100437186889.95839
```

- **Note**: that this implementation of ddof which defaults to 0.
- This argument allows us to set the **degrees of freedom** that we want to use when calculating the variance.
- For example, ddof=0 will allow us to calculate the variance of a **population**.
- Meanwhile, **ddof=1** will allow us to estimate the population variance using a **sample of data**.
- If import the Library as **statistics** module and then call **pvariance()** with our data as an argument.
- That will return the variance of the **population**.
- On the other hand, we can use **Python's variance()** to calculate the variance of a **sample**.
- That's because variance() uses **n - 1** instead of **n** to calculate the variance.

In [576]:

```
def vrnc(x,ddof=0):
    mean_ = sum(x) / len(x)
    dev = [(obj_i - mean_) ** 2 for obj_i in x]
    var = sum(dev) / (len(x)-ddof)
    return var

x = df['Annual_HH_Income']
vrnc(x,ddof=1)
```

Out[576]:

```
102486925397.91666
```

In [577]:

```
Anl_incm_variance = st.variance(df['Annual_HH_Income'])
Anl_incm_variance
```

Out[577]:

```
102486925397.91672
```

Standard Deviation:

- The square root of Variance is called the Standard Deviation.
- It gives Distribution of Variables
- Denoted as

$$\sigma$$

- **Note**: **Range** is **Effected** when Outliers Imputed.
- **Inter Quartile Range** is **NOt Effected** when Outliers Imputed.
- **varience** is **Effected** when Outliers Imputed.
- **Standarad Deviation is Effected** when Outliers Imputed.
- **Measures of Spread** and **Measures of Central Tendency** Both are performs Uni Variable Analysis.

Formula:

- Simply we Formulaed Standard Deviation over population as

$$(\sigma) = \sqrt{Variance} \quad \approx \quad \sqrt{\left\{ \frac{\Sigma_{i=1}^{n} |\mu - obj_i|}{N} \right\}}$$

- Standard Deviation over Sample as

$$(S) = \sqrt{Variance} \quad \approx \quad \sqrt{\left\{ \frac{\Sigma_{i=1}^{n} |\mu - obj_i|}{N-1} \right\}}$$

Example from Scratch and by using Libraries

In [578]:

```
# Std of Papulation

def std_(x):
    mean_ = sum(x) / len(x)
    dev = [(obj_i - mean_) ** 2 for obj_i in x]
    var = sum(dev) / len(x)
    stdev = np.sqrt(var)
    return stdev

x = df['Annual_HH_Income']
std_(x)
```

Out[578]:

```
316918.26531451027
```

In [579]:

```
# Std of Papulation

Anl_incm_std = np.std(df['Annual_HH_Income'])
Anl_incm_std
```

Out[579]:

```
316918.26531451027
```

In [580]:

```
# Std of Sample

def std_(x,ddof = 0):
    mean_ = sum(x) / len(x)
    dev = [(obj_i - mean_) ** 2 for obj_i in x]
    var = sum(dev) / (len(x)-ddof)
    stdev = np.sqrt(var)
    return stdev

x = df['Annual_HH_Income']
std_(x,ddof = 1)
```

Out[580]:

```
320135.79212252516
```

In [581]:

```
# Std of Sample

Anl_incm_std = np.std(df['Annual_HH_Income'],ddof = 1)
Anl_incm_std
```

Out[581]:

```
320135.79212252516
```

In [582]:

```
# Std of Sample

df['Annual_HH_Income'].std()
```

Out[582]:

```
320135.79212252516
```

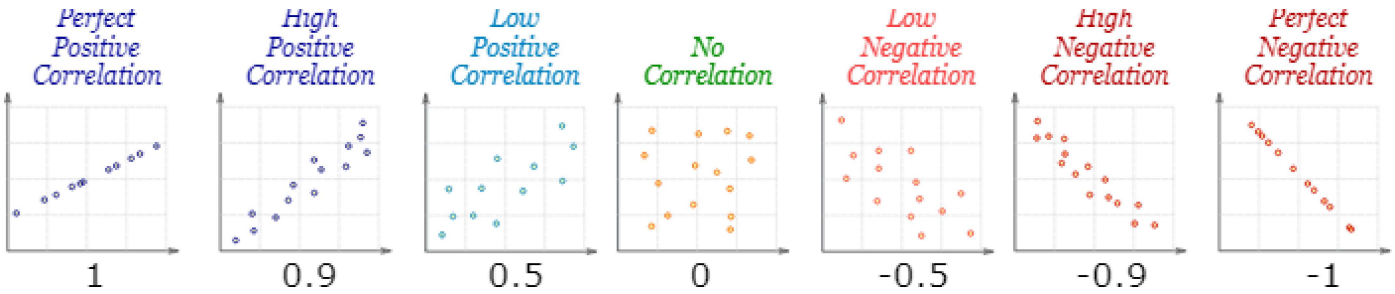
Measure of Relation:

- statistical measures which show a relationship between two or more variables or two or more sets of data.
- Simply find the relationship between two Variables, Known as BiVariables.
- measure of relation broadly categorised as teo types.....
- Covarieance (Having only relation(Direction on Graph))
- Corelation (Having Strength and relation(Direction on Graph))

Correlation:

- Correlation is a statistical measure, Correlation coefficients are used to measure the strength of the relationship between two variables.

- Pearson correlation is the one most commonly used in statistics. This measures the strength and direction of a linear relationship between two variables
- Simplt we can say the degree to which two variables move in a relation to each other.
- In other words it calls as Pearson correlation coefficent.
- Correlation is a term that is a measure of the strength of a linear relationship between two quantitative variables.
- Covariance just measures the relation not Strength.



Formula:

$$Correlation\ \ value\ \ always\ \ lies\ \ in\ \ between\ \ \ (-1 \leq \rho_{x,y} \leq +1)$$
$$Majorly\ \ we\ \ used\ \ Pearson\ \ corelationcoef\,ficent\ \ denoted\ \ by\ \ \ \rho_{x,y}$$

$$Corelation(\rho_{x,y}) = \frac{COV(X,Y)}{\sigma_X * \sigma_Y}$$

In [583]:

```
#Finding Correlation by using Numpy module

x = df.Annual_HH_Income
y = df.Mthly_HH_Income

Corltn_1 = np.corrcoef(x,y)                # Defaultly Pearson Correlation
print(Corltn_1)
print(Corltn_1[0,1])
print((Corltn_1[1,1]))
```

```
[[1.         0.97031542]
 [0.97031542 1.        ]]
0.970315416660371
1.0
```

In [584]:

```
#Finding Correlation by using SciPy module

Corltn_2 = stats.pearsonr(x, y)
print(Corltn_2)
```

```
(0.9703154166603705, 3.028362395598949e-31)
```

In [585]:

```
#Finding Correlation by using Pandas module

Corltn_x = x.corr(y)                          # Defaultly Pearson Correlation
print(Corltn_x)
Corltn_y = y.corr(x)
print(Corltn_y)
Corltn_y = y.corr(y)
print(Corltn_y)
Corltn_x = x.corr(x)
print(Corltn_x)
```

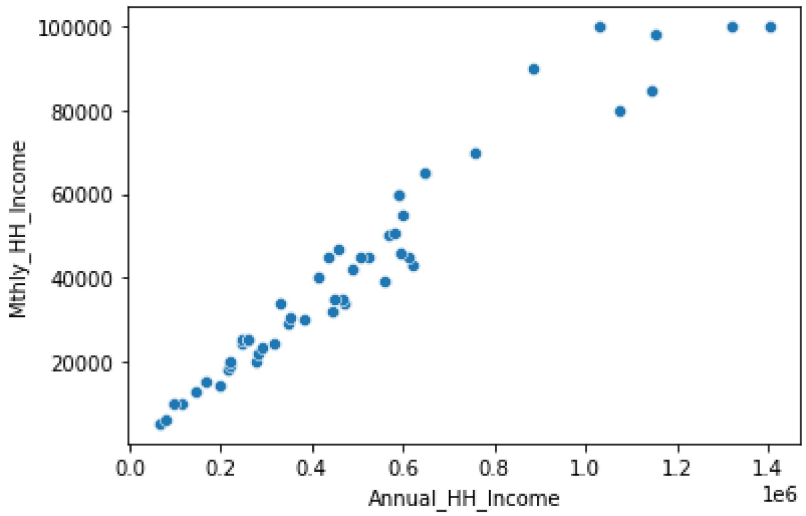
```
0.970315416660371
0.970315416660371
1.0
0.9999999999999998
```

In [586]:

```
d = sns.scatterplot(x='Annual_HH_Income', y='Mthly_HH_Income', data=df)
d
```

Out[586]:

```
<AxesSubplot:xlabel='Annual_HH_Income', ylabel='Mthly_HH_Income'>
```



In [587]:

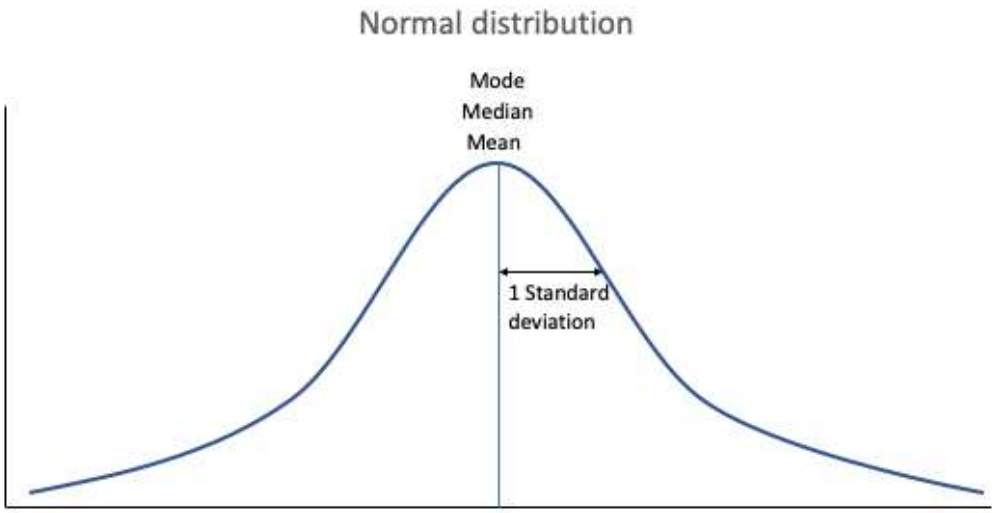
```
viz_1 = df.corr().style.background_gradient(cmap="YlOrBr")
viz_1
```

Out[587]:

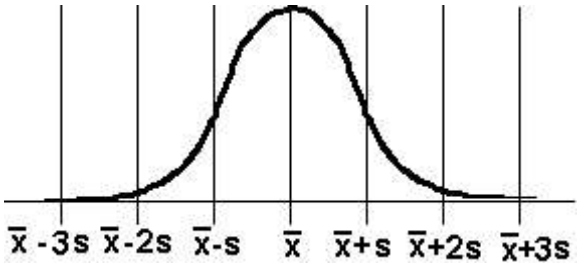
	Mthly_HH_Income	Mthly_HH_Expense	No_of_Fly_Members	Emi_or_Re
Mthly_HH_Income	1.000000	0.649215	0.448317	0.
Mthly_HH_Expense	0.649215	1.000000	0.639702	0.
No_of_Fly_Members	0.448317	0.639702	1.000000	0.
Emi_or_Rent_Amt	0.036976	0.405280	0.085808	1.
Annual_HH_Income	0.970315	0.591222	0.430868	0.
No_of_Earning_Members	0.347883	0.311915	0.597482	-0.

Normal Distribution:

- Normal 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.**
- it also known as the **Gaussian distribution**.
- statistically normal distribution has the mean as its **mean**, its **mode**, and its **median** and it looks like a **symmetric mound or bell.**
- the word normal is a very powerful adjective when used to describe a frequency distribution or when used to describe the data of a sample or population.

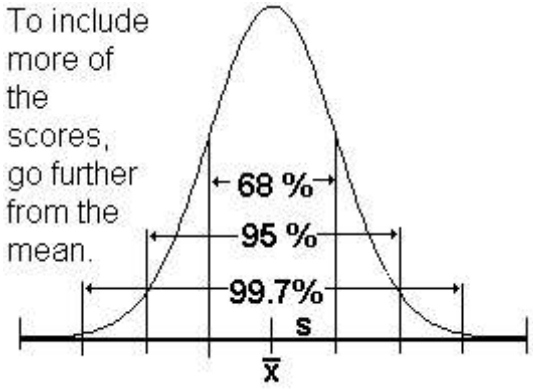


The Data Is Centered About the **Mean-Mode-Median**



Empirical rule

- The empirical rule, or the 68-95-99.7 rule, tells us where most of values lie in a normal distribution:
- Around 68% of values are within 1 standard deviation from the mean.
- Around 95% of values are within 2 standard deviations from the mean.
- Around 99.7% of values are within 3 standard deviations from the mean.



Feature of Normal Distribution:

- The mean-mode-median is in the center.
- It is the mean because it is the ARITHMETIC average of all the scores.
- It is the mode because of all the scores the mean score happens MOST often.
- It is the median because when the scores are displayed from lowest to highest, the mean is the MIDDLE score, the median.
- The EXPECTED value is the mean.
- The frequency curve is bell shaped.
- The bell shape has perfect bilateral symmetry - the left balances exactly with the right.
- The score at -2 is balanced by a score at +2 and the frequencies from 0 to +2 and from 0 to -2 are equal.
- The area under the curve from 0 to +2 is exactly the same as the area under the curve from 0 to -2.
- Fifty percent of the scores are above the mean and 50% are below the mean.
- The probability a score is above the mean is 50% and the probability a score is below the mean is 50%.
- Most of the scores are in **the middle, about the mean, and few are in the tails, at the extremes.**

Axiom's of Normal Distribution:

- The area under the curve is equal to 1.
- The probability of an event that does not happen is 0.
- The sum of the probabilities of all events is 1.
- The standard deviation tells one how the scores are spread out and therefore the fatness or skinniness of the bell.

Formula:

- **Once we have the mean and standard deviation of a normal distribution, we can fit a normal curve to data using a probability density function.**

$$f(X) = \frac{1}{\sigma\sqrt{2\pi}} e^{-} \frac{(X-\mu)^2}{2\sigma^2}$$

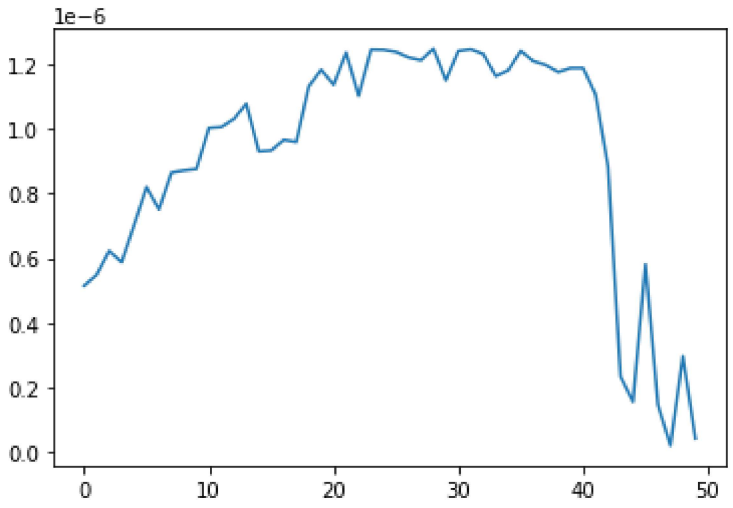
- Wher f(x) = probability
- x = value of the variable
- μ = mean
- σ = standard deviation
- σ2 = variance

In [588]:

```
x = norm.pdf(df['Annual_HH_Income'], Anl_incm_mean, Anl_incm_std)
plt.plot(x)
```

Out[588]:

```
[<matplotlib.lines.Line2D at 0x23011645730>]
```

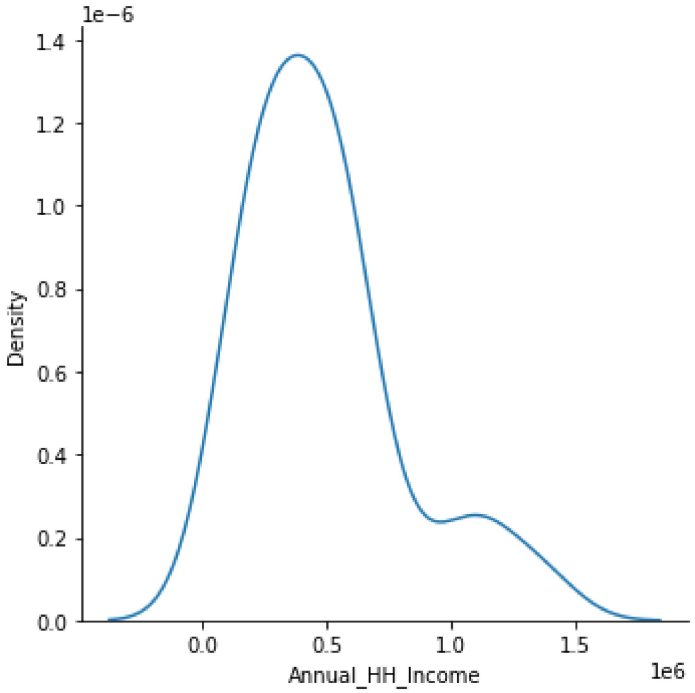


In [589]:

```
sns.displot(df, x='Annual_HH_Income', kind="kde")
```

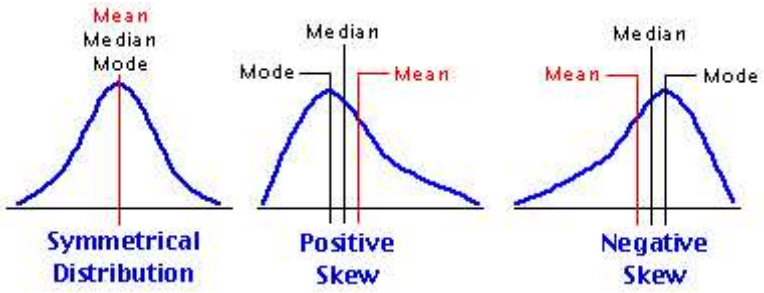
Out[589]:

```
<seaborn.axisgrid.FacetGrid at 0x23012887a30>
```



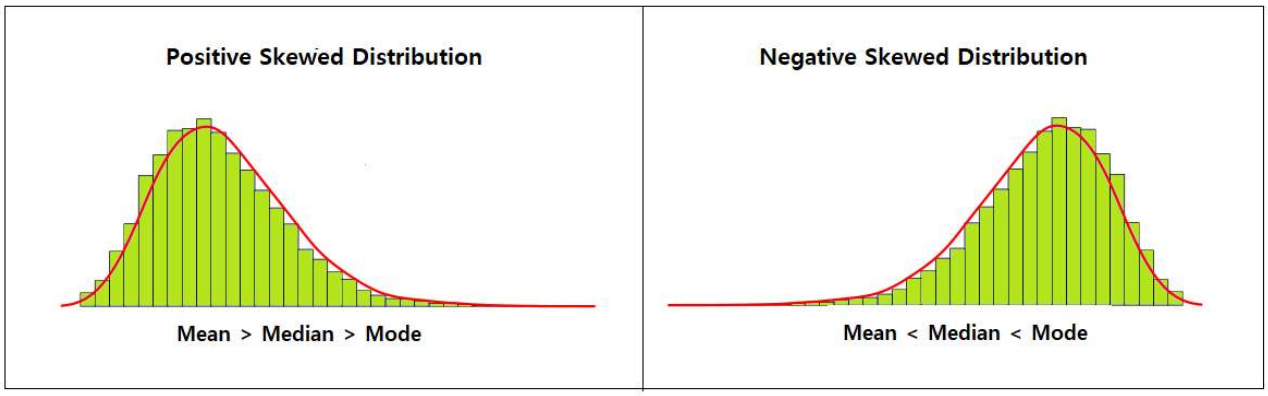
Positively Skewed & Negatively Skewed Normal Distribution:

- Skewness refers to a distortion or asymmetry that deviates from the symmetrical bell curve, or normal distribution, in a set of data.
- If the curve is shifted **to the left or to the right**, it is said to be skewed.
- Skewness can be quantified as a representation of the extent to which a given distribution varies from a normal distribution. A normal distribution has a skew of **zero**.
- Skewness value lies in between **(-0.5 to +0.5)we called it as "Symmetric".**
- If the skewness of variable is **less than -0.5 or morethan +0.5 it is not Symmetric.**



Effect on Mean, Median and Mode due to Skewness

- Right-skewed distributions are also called **positive-skew distributions**.
- That's because there is a long tail in the positive direction on the number line.
- The mean is also to the right of the peak.
- The normal distribution is the most common distribution you'll come across.
- 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.
- Besides positive and negative skew, distributions can also be said to have **zero or undefined skew.**
- In the curve of a distribution, the data on the right side of the curve may taper differently from the data on the left side.
- These taperings are known as "tails." Negative skew refers to a longer or fatter tail on the left side of the distribution, while positive skew refers to a longer or fatter tail on the right.
- The mean of positively skewed data will be greater than the median.
- In a distribution that is negatively skewed, the exact opposite is the case: the mean of negatively skewed data will be less than the median.
- If the data graphs symmetrically, the distribution has zero skewness.



Formula:

- There are several ways to measure skewness. Pearson's first and second coefficients of skewness are two common ones.
- Pearson's first coefficient of skewness, or Pearson mode skewness, subtracts the mode from the mean and divides the difference by the standard deviation.
- Pearson's second coefficient of skewness, or Pearson median skewness, subtracts the median from the mean, multiplies the difference by three, and divides the product by the standard deviation.
- The Formulae for Pearson's Skewness Are:

$$Sk_1 = \frac{\mu - Mo}{s}$$

$$Sk_2 = \frac{3\mu - Md}{s}$$

- **Where.....**
- Sk_1 is the Pearson's first coefficient of skewness and Sk2 is the second
- s=the standard deviation for the sample
- \mu =is the mean value
- Mo=the modal (mode) value
- Md=is the median value

Note:

- skewness when judging a return distribution because it, like **kurtosis**, considers the extremes of the data set rather than focusing solely on the average.
- Short- and medium-term investors in particular need to look at extremes because they are less likely to hold a position long enough to be confident that the average will work itself out.
- Investors commonly use standard deviation to predict future returns, but the standard deviation assumes a normal distribution.
- As few return distributions come close to normal, skewness is a better measure on which to base performance predictions.
- **This is due to skewness risk.**
- Skewness risk is the increased risk of turning up a data point of high skewness in a skewed distribution.
- Many financial models that attempt to predict the future performance of an asset assume a normal distribution, in which measures of central tendency are equal.
- If the data are skewed, this kind of model will always underestimate skewness risk in its predictions.

In [590]:

```
a = skew(df['Annual_HH_Income'])
a
```

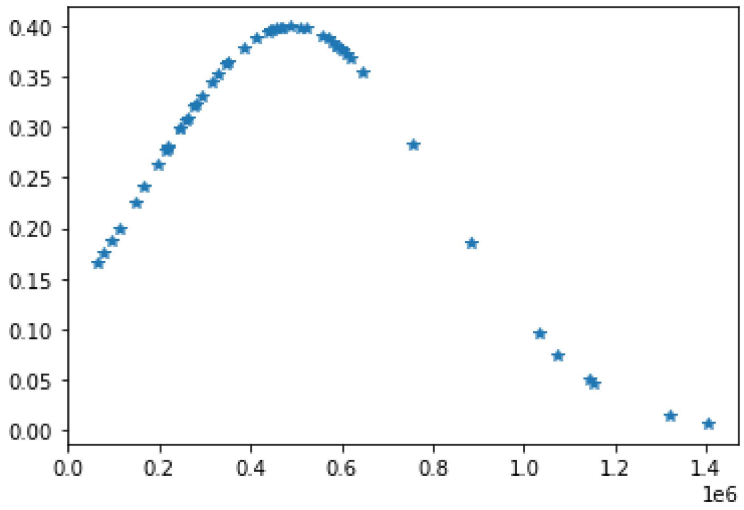
Out[590]:

```
1.156858143785799
```

In [591]:

```
A = df['Annual_HH_Income'].mean()

std_S = np.std(df['Annual_HH_Income'], ddof = 1)

d = df['Annual_HH_Income']

sk = (1/(np.sqrt(2.*np.pi)))* np.exp(-((d-A)**2)/(2*std_S**2))

p.plot(d, sk, '*')


print( '\nSkewness of d : ', skew(sk))
```

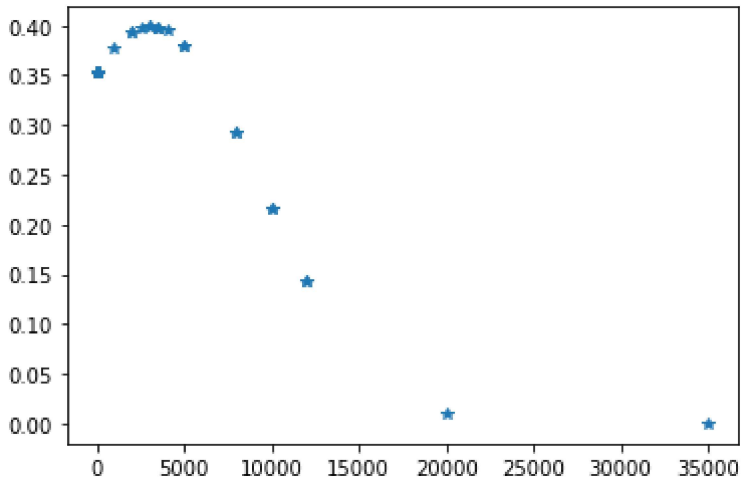
```
Skewness of d :  -1.1778312064829035
```



In [592]:

```
A = df['Emi_or_Rent_Amt'].mean()

std_S = np.std(df['Emi_or_Rent_Amt'], ddof = 1)

d = df['Emi_or_Rent_Amt']

sk = (1/(np.sqrt(2.*np.pi)))* np.exp(-((d-A)**2)/(2*std_S**2))

p.plot(d, sk, '*')


print( '\nSkewness of d : ', skew(sk))
```

```
Skewness of d :  -2.604074653745475
```

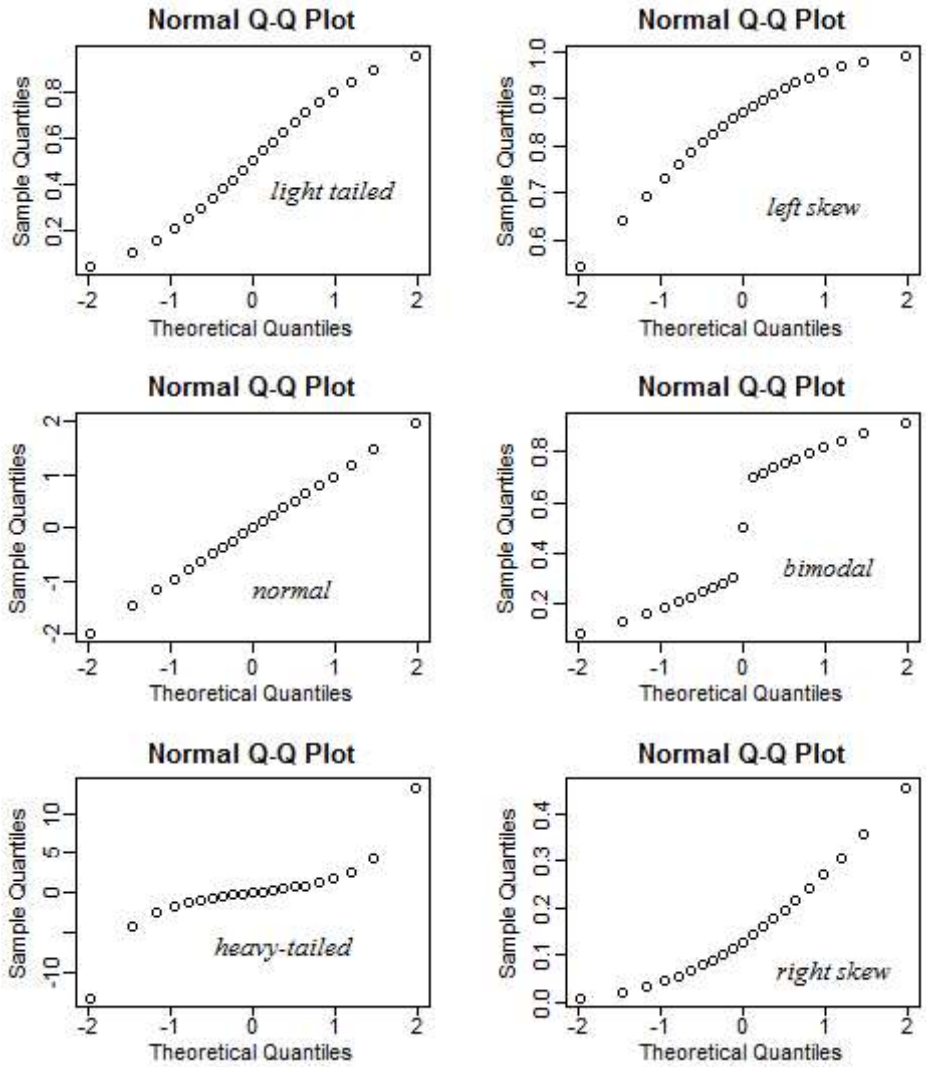


Explaination of Quantile-Quantile(QQ) Plot:

- The Q-Q plot, or quantile-quantile plot, is a graphical tool to help us assess if a set of data plausibly came from some theoretical distribution such as a Normal or exponential.
- For example, if we run a statistical analysis that assumes our dependent variable is Normally distributed, we can use a Normal Q-Q plot to check that assumption.
- It's just a visual check, not an air-tight proof, so it is somewhat subjective.
- But it allows us to see at-a-glance if our assumption is plausible, and if not, how the assumption is violated and what data points contribute to the violation.
- A Q-Q plot is a scatterplot created by plotting two sets of quantiles against one another.
- If both sets of quantiles came from the same distribution, we should see the points forming a line that's roughly straight.
- These are often referred to as "percentiles".
- These are points in your data below which a certain proportion of your data fall.
- For example, imagine the classic bell-curve standard Normal distribution with a mean of 0.
- The 0.5 quantile, or 50th percentile, is 0.
- Half the data lie below 0. That's the peak of the hump in the curve.
- The 0.95 quantile, or 95th percentile, is about 1.64.
- 95 percent of the data lie below 1.64.

Implementation of Quantile-Quantile(QQ) Plot: :

- **To Check the given variable is normally distributed or not, ie (Normal or Not) we used a test called as QQ Plot.**



- 1. if we plot a graph between two different variables, all the data points lying on a same line.

$$(line = 45^0)$$

  Then we called as the distribution is Normal Distribution.
- 2. In case ther is any distrabances like data points distributed away from line then we say they are not distributed as Normal.
- 3. first of all we have to check.

$$I.\ Therotical\ \ Quantity\ \ (Normal\ \ Distribution\ \ mean = 0,\ \ std = 1),$$
$$II.\ \ Observed\ \ Quantity\ \ (Given\ \ Distribution\ \ or\ \ Column).$$

- 4. sort the values of Therotical Quantity and Observed Quantity.
- 5. After sorting plot a Scatterplot and take oserved Quantity, if all the data points lying in a 45 degree line then we can say Normally distributed Otherwise Not distributed Normally.
  Note: Theoretical Quantities always lying Normal Distribution because we use the"np.random.normal(loc=0, scale=1)"

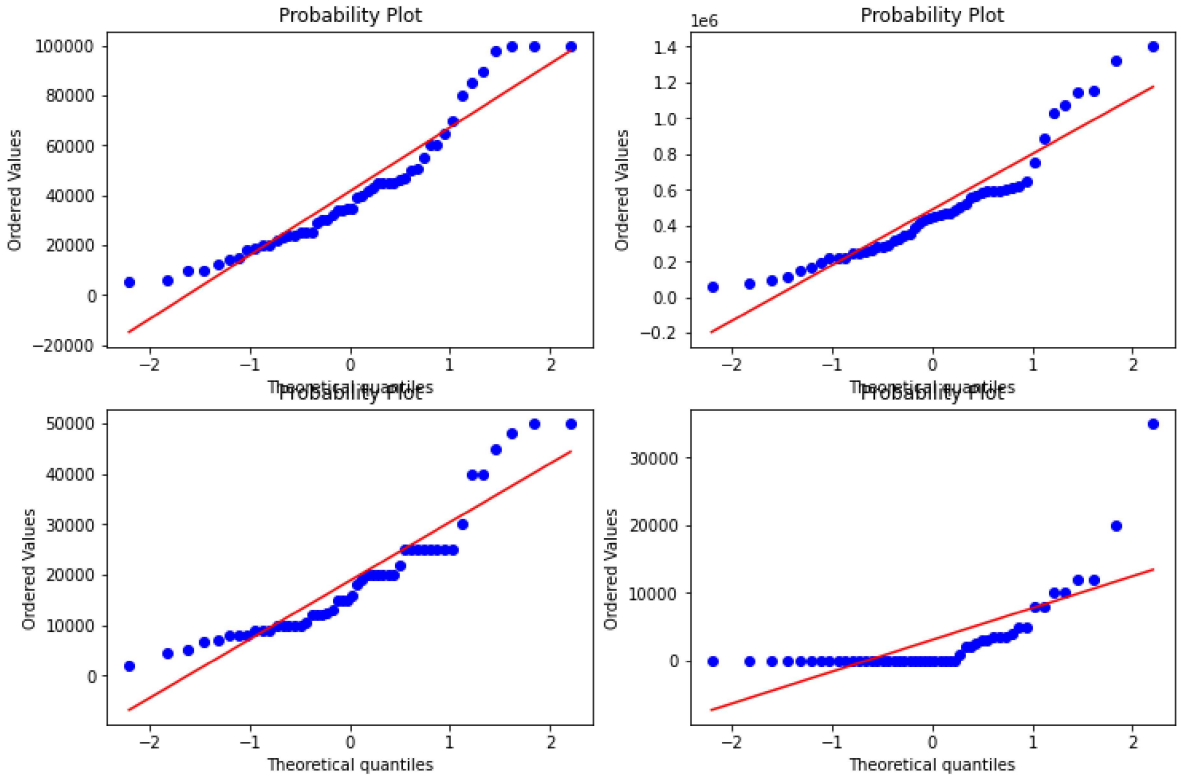
In [593]:

```
df_1 = df[['Mthly_HH_Income', 'Annual_HH_Income']]
df_2 = df[['Mthly_HH_Expense', 'Emi_or_Rent_Amt']]

#Checking Skewness and Kurtosis

print('*'*10, 'SKEWNESS', '*'*10)
print()
print('Monthly Incom', df['Mthly_HH_Income'].skew())
print('Anual Incom', df['Annual_HH_Income'].skew())
print('Monthly Expenses', df['Mthly_HH_Expense'].skew())
print('EMI  & Rent', df['Emi_or_Rent_Amt'].skew())
print('*'*30)
print()
print()
print('*'*10, 'KURTOSIS', '*'*10)
print()
print('Monthly Incom', df['Mthly_HH_Income'].kurt())
print('Anual Incom', df['Annual_HH_Income'].kurt())
print('Monthly Expenses', df['Mthly_HH_Expense'].kurt())
print('EMI  & Rent', df['Emi_or_Rent_Amt'].kurt())
print('*'*30)
```

```
********** SKEWNESS **********

Monthly Incom 0.9246148763777229
Anual Incom 1.1929490975818218
Monthly Expenses 1.1994608030097127
EMI  & Rent 3.403679844103547
******************************


********** KURTOSIS **********

Monthly Incom 0.11555007146606489
Anual Incom 1.1012908548129197
Monthly Expenses 0.9424897886350738
EMI  & Rent 14.202522820819258
******************************
```

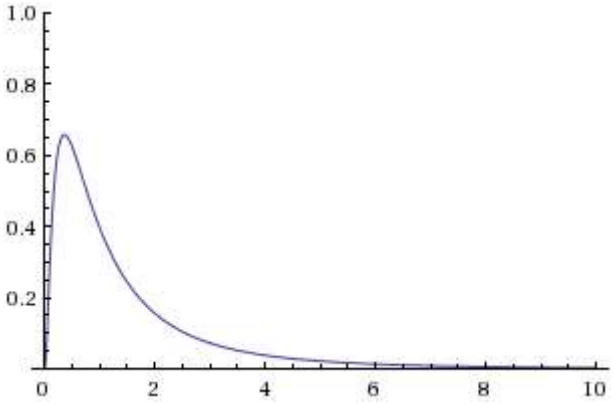
In [594]:

```
plt.figure(figsize=(12, 8))

# Creating a plot with 1 row and 2 cols

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

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

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

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



plt.show()
```

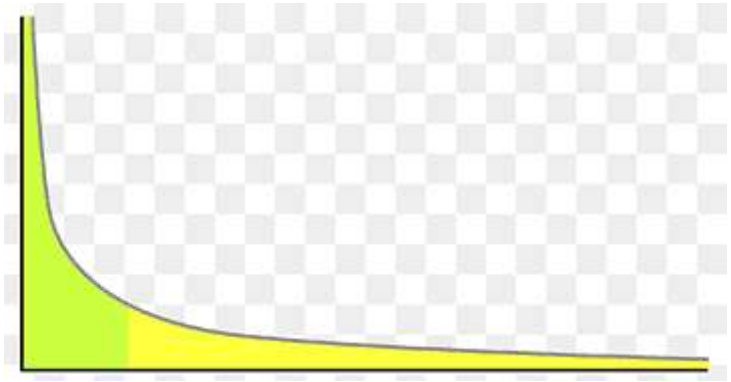


Explaination of Box Cox:

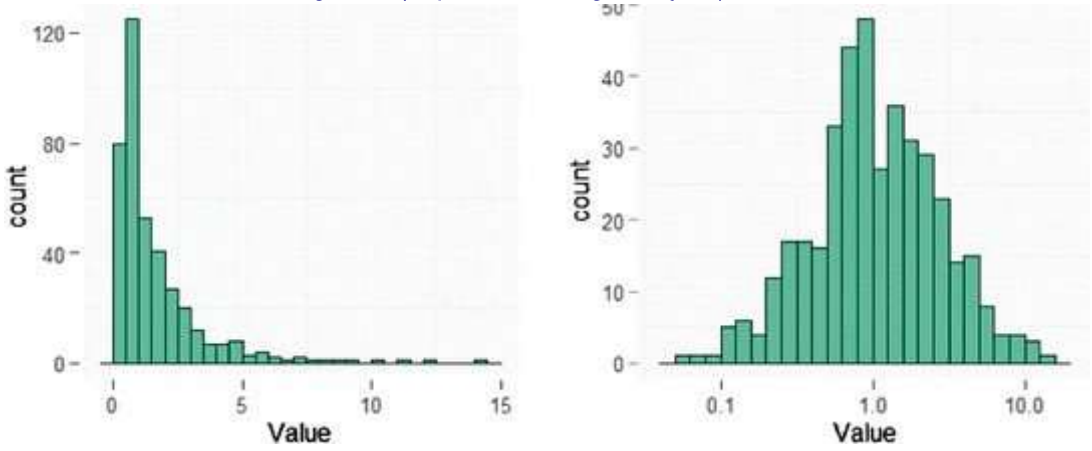
- The Box-Cox is a transformation technique, which transforms data so that it closely resembles a normal distribution.
- In many statistical techniques, we assume that the errors are normally distributed.
- This assumption allows us to construct confidence intervals and conduct hypothesis tests.
- Simply we can say Box Cox is a transformation of non-normal dependent variables into a normal shape.



- Image_1



- Image_2
- Image_1, Image_2 represent **Log Normal, Pareto Distribution** Respectivly.
- if we observe this two distributions it contains more outliers, the magnitude of outlier is very high.
- So it is imposible to delete the column because it contain lot of data.
- if we delete then whole data will goned.(ie predict wrong Analysis).



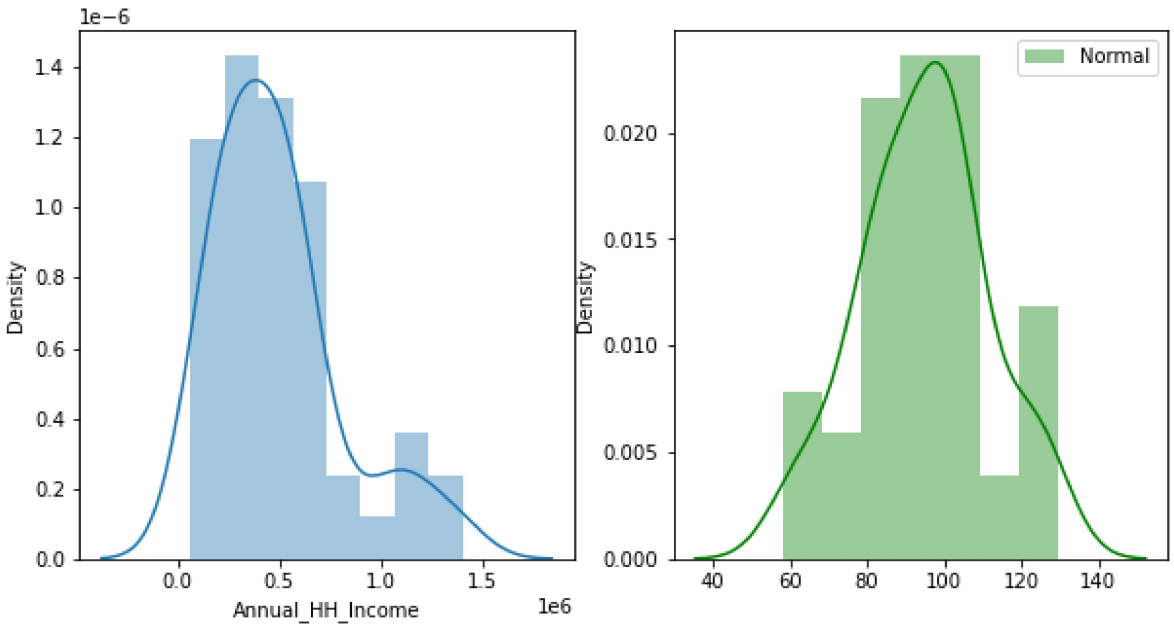
- **Before transformation plot looks like Non-Normal Distribution after appling BoxCox Transformation Distribution transforms into Normal.**

Implementation of Box Cox::

In [595]:

```
data = df.Annual_HH_Income

# transform training data & save lambda value
fitted_data, fitted_lambda = stats.boxcox(data)

# creating axes to draw plots
fig, ax = plt.subplots(1, 2)

# plotting the original data(non-normal) and
# fitted data (normal)
sns.distplot(data, label = "Non-Normal", ax = ax[0])

sns.distplot(fitted_data, label = "Normal", color ="green", ax = ax[1])

# adding legends to the subplots
plt.legend(loc = "upper right")

# rescaling the subplots
fig.set_figheight(5)
fig.set_figwidth(10)

print(f"Lambda value used for Transformation: {fitted_lambda}")
```

```
Lambda value used for Transformation: 0.24689401204475175
```



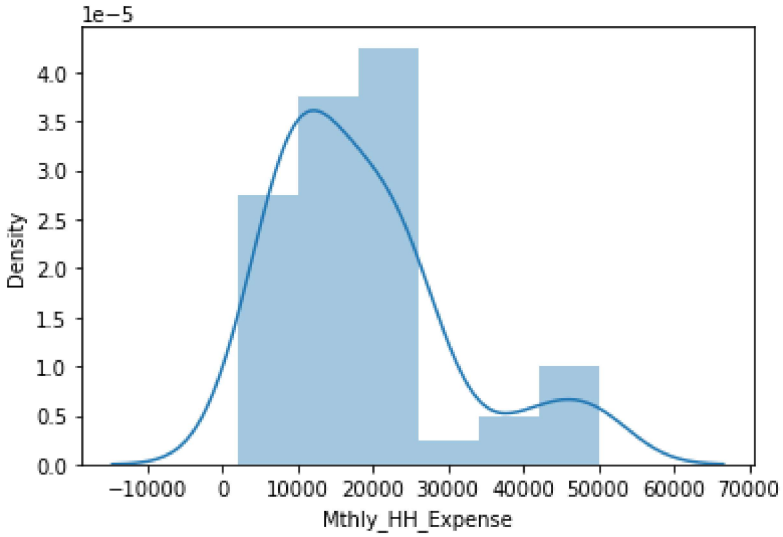
- **Before transformation plot looks like Non-Normal Distribution after appling BoxCox Transformation Distribution transforms into Normal.**

In [596]:

```
rv = df.Mthly_HH_Expense

sns.distplot(rv)
```

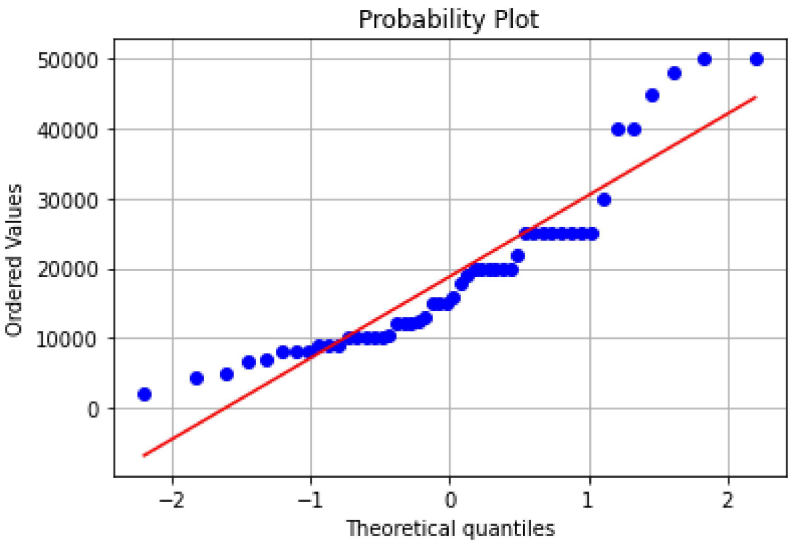
Out[596]:

```
<AxesSubplot:xlabel='Mthly_HH_Expense', ylabel='Density'>
```



In [597]:

```
# Normality Test

stats.probplot(rv, dist="norm", plot=plt)

plt.grid()
```



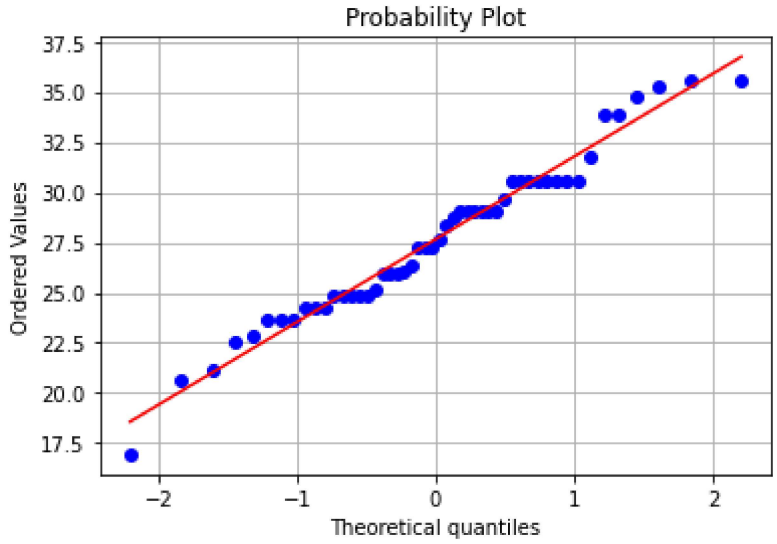
In [598]:

```
# x = tranformed by box-cox, l = Lambda

x, l = stats.boxcox(rv)

print(l)
```

```
0.18883043817938627
```

In [599]:

```
stats.probplot(x, dist="norm", plot=plt)

plt.grid()
```



In [ ]: