Innomatics Research Labs

SHUBHAM SALOKHE

Descriptive Statistics and Python Implementation

- Mean
- Median
- Mode
- Variance
- Standard Deviation
- Correlation
- Normal Distribution (use references)
- Feature of Normal Distribution
- Positively Skewed & Negatively Skewed Normal Distribution
- Effect on Mean, Median and Mode due to Skewness
- Explain QQ Plot and show the implementation of the same
- Explain Box Cox and show the implementation of the same

In [1]:
```
# importing pandas libraries for importing data
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
```

In [2]:
```
df = pd.read_csv("data.csv")
df.head(10)
```

Out[2]:

	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
5	14000	8000	2	0	19656
6	15000	16000	3	35000	16740
7	18000	20000	5	8000	21600
8	19000	9000	2	0	21888
9	20000	9000	4	0	22080

Descriptive Statistics.

1. Mean

Mean is the sum of the all values, devided by the number of the values

In [3]:
```
list_num = [9,3,1,8,3,6]
print("Values: {}".format(list_num))
print("Number of values: {}".format(len(list_num)))
```

```
Values: [9, 3, 1, 8, 3, 6]
Number of values: 6
```

In [4]:
```
# making the sum of all numbers
sum_num = (9+3+1+8+3+6)

print("Sum of all Values: {}".format(sum_num))
```

```
Sum of all Values: 30
```

Mean Formula

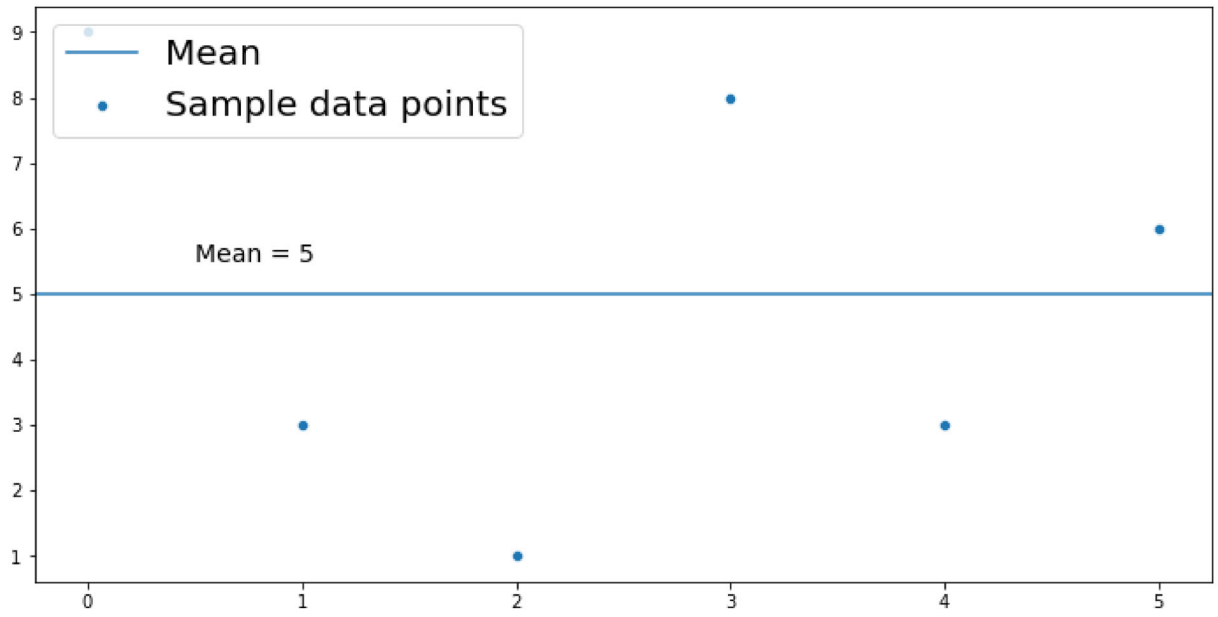
$$\bar{X} = \left(\frac{\sum_{i=0}^{N} X_i}{N} \right)$$

In [5]:
```
Mean = sum_num/6
print("Mean: {}".format(Mean))
```

```
Mean: 5.0
```

In [6]:
```
plt.figure(figsize =(12,6))
graph =sns.scatterplot(y=list_num , x = range(len(list_num)), label = "Sample dat
graph.axhline(5, label="Mean")
plt.text(0.5, 5.5, 'Mean = 5', size ='14')
plt.legend(loc="upper left", fontsize = 20 )
plt.legend
```

Out[6]:
```
<function matplotlib.pyplot.legend(*args, **kwargs)>
```



Similarly. we do with our data set, we are going to make mean of Mthly_HH_Income column

In [7]:
```
list_nums = df['Mthly_HH_Income'].values
print("Values: {}".format(list_nums))
print("Number of values: {}".format(len(list_nums)))
```

```
Values: [  5000   6000  10000  10000  12500  14000  15000  18000  19000  20000
  20000  22000  23400  24000  24000  25000  25000  25000  29000  30000
  30500  32000  34000  34000  35000  35000  39000  40000  42000  43000
  45000  45000  45000  45000  46000  47000  50000  50500  55000  60000
  60000  65000  70000  80000  85000  90000  98000 100000 100000 100000]
Number of values: 50
```

In [8]:
```
# making the sum of all numbers
sum_nums = sum(list_nums)
# Here i'm using python inbuilt function(sum)

print("Sum of all Values: {}".format(sum_nums))
```

```
Sum of all Values: 2077900
```

In [9]:
```
# Taking out mean
Mean = sum_nums/len(list_nums)
#Mean = 2077900/50
print("Mean of Mthly_HH_Income : {}".format(Mean))
```

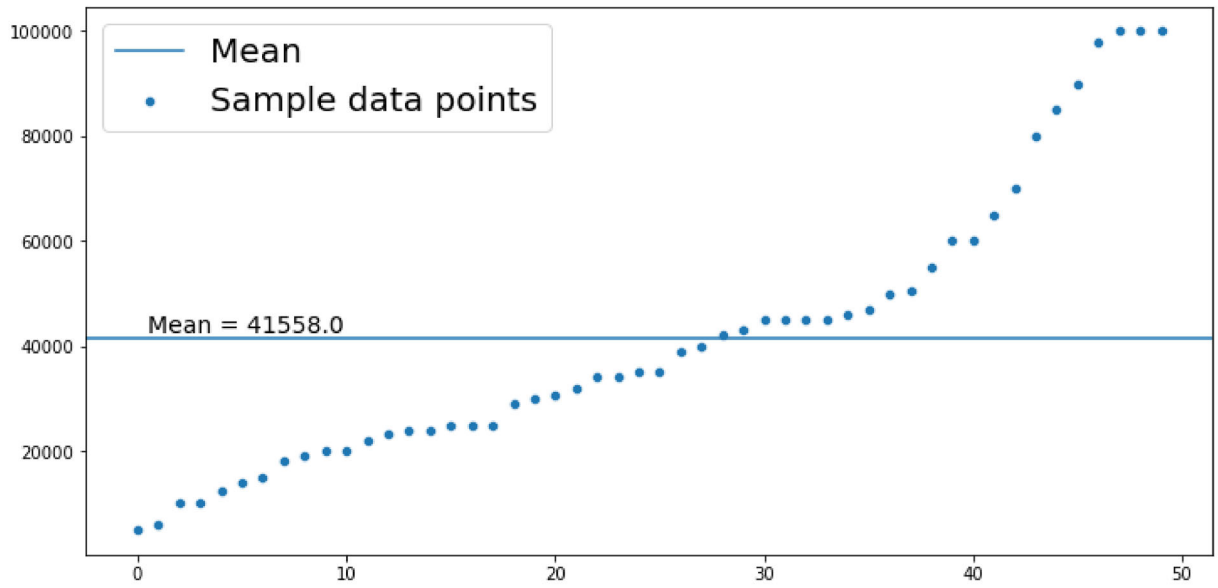
```
Mean of Mthly_HH_Income : 41558.0
```

In [10]:
```
# Similarlly we can use numpy for mean
import numpy
print(numpy.mean(df['Mthly_HH_Income']))
```

```
41558.0
```

In [11]:
```
plt.figure(figsize =(12,6))
graph =sns.scatterplot(y=list_nums , x = range(len(list_nums)), label = "Sample d
graph.axhline(Mean, label="Mean")
plt.text(0.5, Mean+1000, 'Mean = 41558.0', size ='14')
plt.legend(loc="upper left", fontsize = 20 )
plt.legend
```

Out[11]:
```
<function matplotlib.pyplot.legend(*args, **kwargs)>
```



2. Median

The Median is the middle number in the list of numbers ordered from lowest to highest

$$1, 3, 3, \mathbf{6}, 7, 8, 9$$

$$\text{Median} = \underline{\underline{6}}$$

$$1, 2, 3, \mathbf{4}, \mathbf{5}, 6, 8, 9$$

$$\text{Median} = (4 + 5) \div 2$$

$$= \underline{\underline{4.5}}$$

Formula of median

If N is odd

$$Median = \left(\frac{N+1}{2}\right)^{th} Observation$$

If N is even

$$Median = \frac{\left(\frac{N}{2}\right)^{th}\left(\frac{N}{2}+1\right)^{th} Observation}{2}$$

In [12]:
```
# appling on data set, Mthly_HH_Income
# step-1 sorting the values ascending order
n_num = df['Mthly_HH_Income'].sort_values(ascending= True)
print(n_num.values)
```

```
[  5000   6000  10000  10000  12500  14000  15000  18000  19000  20000
  20000  22000  23400  24000  24000  25000  25000  25000  29000  30000
  30500  32000  34000  34000  35000  35000  39000  40000  42000  43000
  45000  45000  45000  45000  46000  47000  50000  50500  55000  60000
  60000  65000  70000  80000  85000  90000  98000 100000 100000 100000]
```

In [13]:
```
# Step 2: counting the values
n = len(n_num)
print( "Even Number:{}".format(n))
```

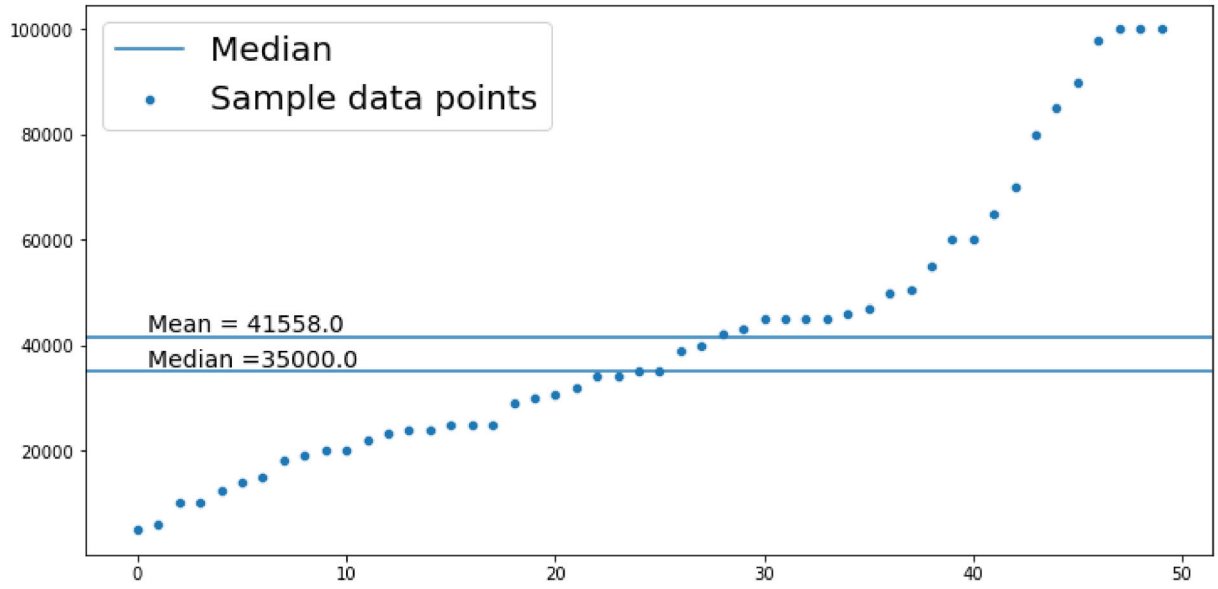
```
Even Number:50
```

In [14]:
```
if n % 2 == 0:
    median1 = n_num[n//2]
    median2 = n_num[n//2 - 1]
    median = (median1 + median2)/2
else:
    median = n_num[n//2]
print("Median is: " + str(median))
```

```
Median is: 35000.0
```

In [15]:
```
plt.figure(figsize =(12,6))
graph =sns.scatterplot(y=list_nums , x = range(len(list_nums)), label = "Sample d
graph.axhline(median, label="Median")
plt.text(0.5,median+1000, 'Median =35000.0', size ='14')
plt.legend(loc="upper left", fontsize = 20 )
graph.axhline(Mean, label="Mean")
plt.text(0.5, Mean+1000, 'Mean = 41558.0', size ='14')
plt.legend
```

Out[15]: <function matplotlib.pyplot.legend(*args, **kwargs)>



3. Mode

The **Mode** is the value that appears most often in a set of data

Formula of Mode

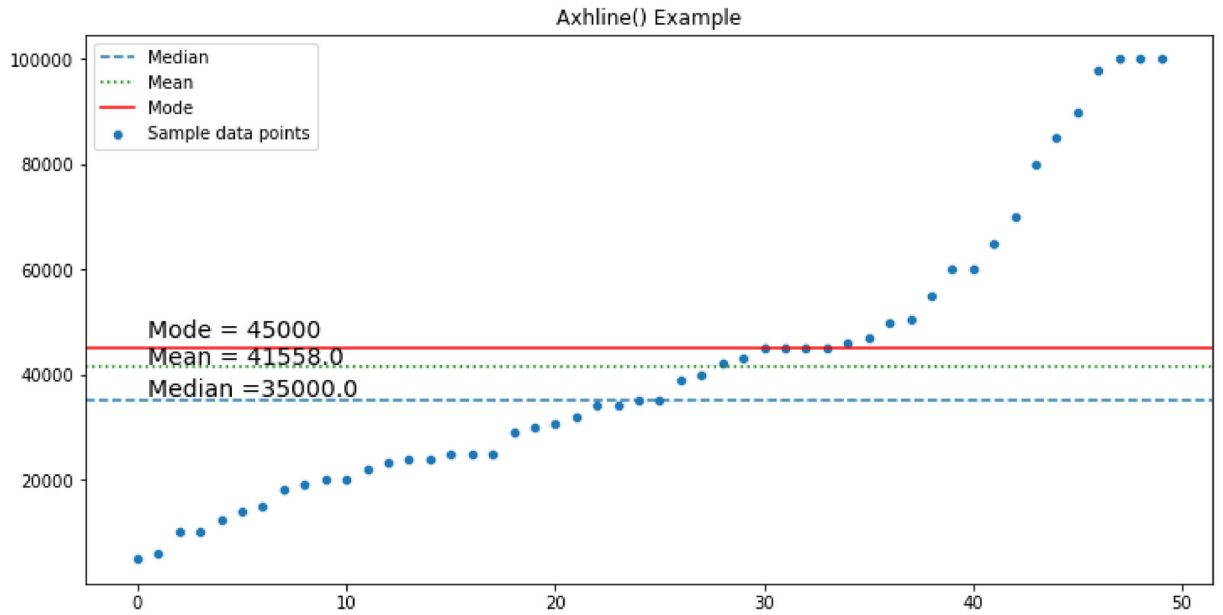
$$M_0 = l + h\left(\frac{f_1 - f_0}{2f_1 - f_0 - f_2}\right)$$

Where,
l = lower limit of the modal class,
h = size of the class interval (assuming all class size to be equal),
f_1 = frequency of the modal class,
f_0 = frequency of the class preceding the modal class,
f_2 = frequency of the class succeding the modal class.

In [16]:
```
from collections import Counter

# list of elements to calculate mode
n_num = df['Mthly_HH_Income']
n = len(n_num)

data = Counter(n_num)
get_mode = dict(data)
mode = [k for k, v in get_mode.items() if v == max(list(data.values()))]

if len(mode) == n:
    get_mode = "No mode found"
else:
    get_mode = "Mode is / are: " + ', '.join(map(str, mode))
```

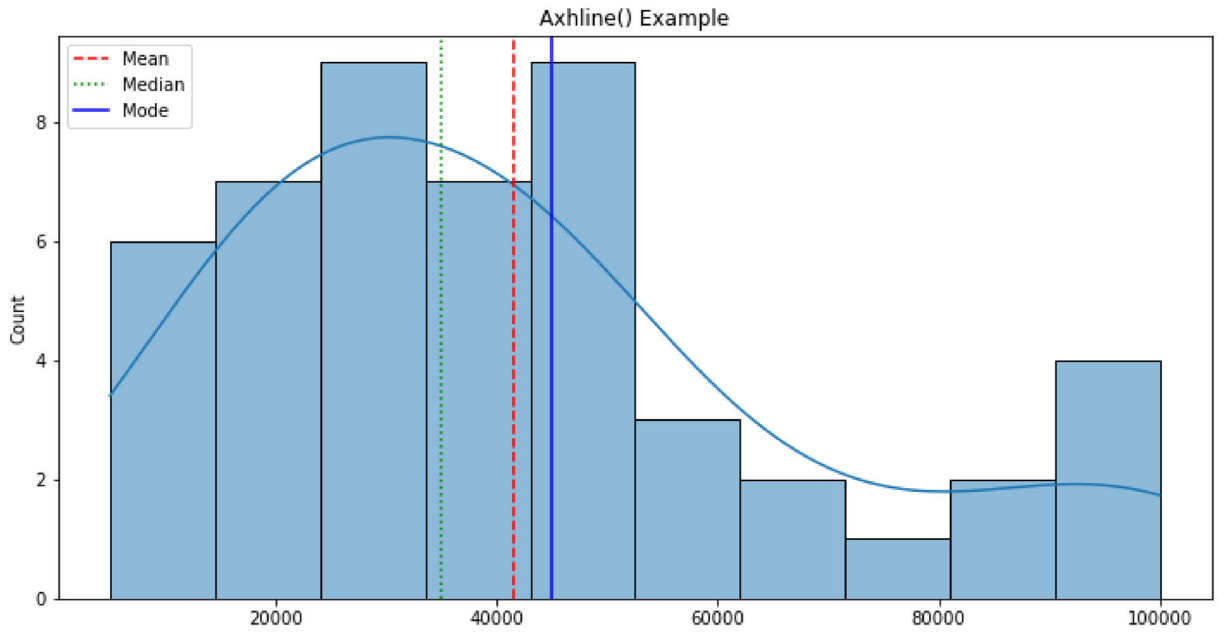
In [17]:
```
plt.figure(figsize =(12,6))
graph =sns.scatterplot(y=list_nums , x = range(len(list_nums)), label = "Sample d
graph.axhline(median, label="Median", linestyle ="--")
plt.text(0.5,median+1000, 'Median =35000.0', size ='14')
plt.legend(loc="upper left", fontsize = 20 )
graph.axhline(Mean, label="Mean",color ="green", linestyle =":")
plt.text(0.5, Mean+500, 'Mean = 41558.0', size ='14')
graph.axhline(mode[0], label="Mode", color ="red")
plt.text(0.5, mode[0]+2000, 'Mode = 45000', size ='14')

plt.title('Axhline() Example')
plt.legend(loc ='upper left')
```

Out[17]: <matplotlib.legend.Legend at 0x2ccc3aa3fa0>



In [18]:
```
plt.figure(figsize =(12,6))
ax_hist =sns.histplot(list_nums, kde=True, bins= 10)
ax_hist.axvline(Mean, color='r', linestyle='--', label="Mean")
ax_hist.axvline(median, color='g', linestyle=':', label="Median")
ax_hist.axvline(mode[0], color='b', linestyle='-', label="Mode")
plt.title('Axhline() Example')
plt.legend(loc ='upper left')
```

Out[18]:
```
<matplotlib.legend.Legend at 0x2ccc4b67940>
```



3. Variance

it's the average squared deviation of the list elements from the average value.`

1. Population Variance,

$$s^2 = \frac{SS}{N} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{N}$$

2. Sample Variance

$$s^2 = \frac{SS}{N-1} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{N-1}$$

where,

SS = sum of squared errors

N = number of observations in the group

x_i = i^{th} score in a group

$\bar{x}$ = mean of the group.

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

In [19]:
```
# 2. W/O External Dependency
lst= df['Mthly_HH_Income']
avg = sum(lst) / len(lst) # mean of gorup
var = sum((x-avg)**2 for x in lst) / (len(lst)-1)
print("Variance: {}".format(var))
```

```
Variance: 681100853.0612245
```

In [20]:
```
import statistics

# Creating a sample of data
sample = df['Mthly_HH_Income']

# Prints variance of the sample set

# Function will automatically calculate
# it's mean and set it as xbar
print("Variance of sample set is % s"
      %(statistics.variance(sample)))
```

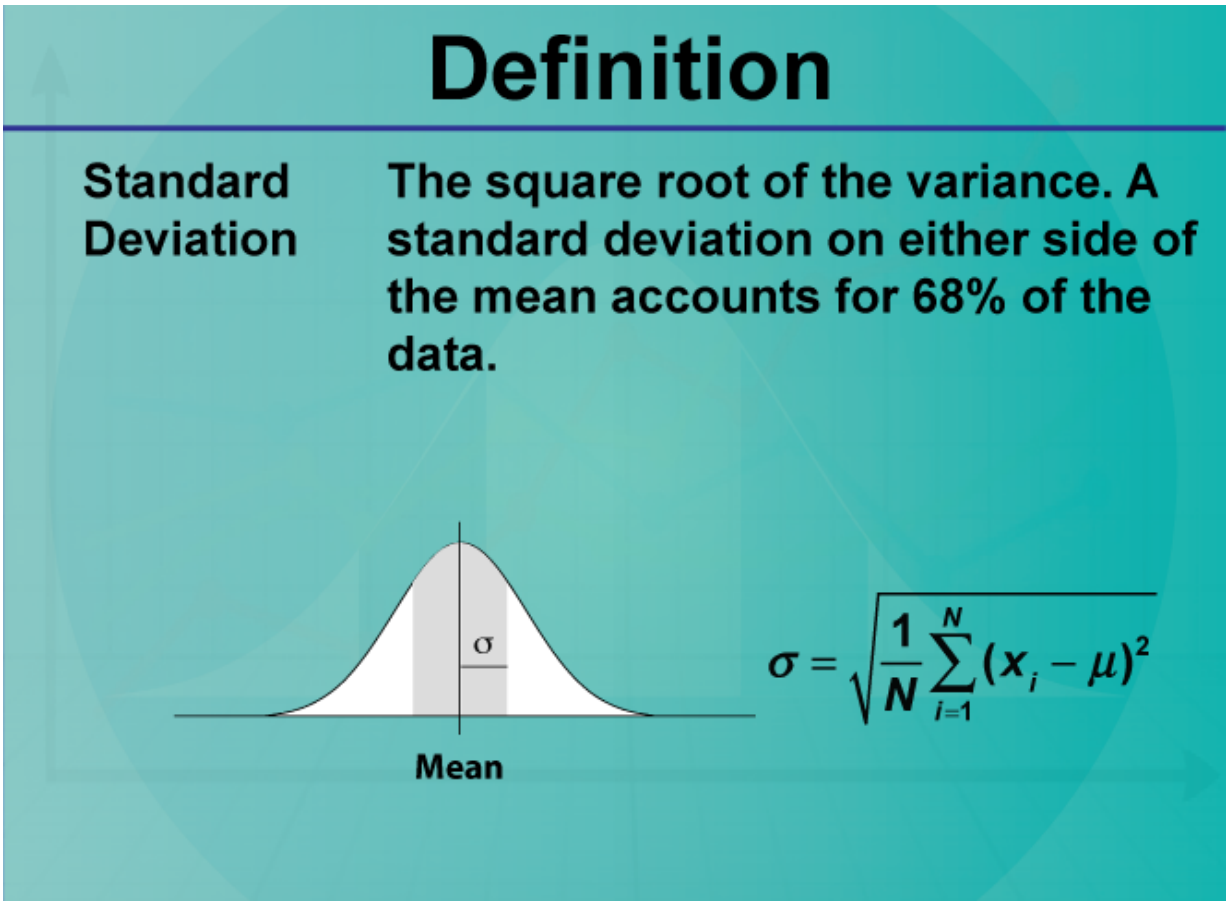
```
Variance of sample set is 681100853.0612245
```

4. Standard Deviation

In [21]:
```
df.std()
```

```
<ipython-input-21-ce97bb7eaef8>:1: FutureWarning: Dropping of nuisance columns
in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future v
ersion this will raise TypeError.  Select only valid columns before calling the
reduction.
  df.std()
```

Out[21]:
```
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
```



$$s = \sqrt{s^2} = \sqrt{\frac{SS}{N-1}} = \sqrt{\frac{\sum(x_i - \bar{x})^2}{N-1}}$$

where,
S^2*is the variance
SS is the sum of squared errors
N is the number of observations in the group
$\mathbf{x_i}$ is the $\mathbf{i^{th}}$ score in a group; and
$\bar{x}$ is the mean of the group.

In [22]:
```
def get_std_dev(ls):
    n = len(ls)
    mean = sum(ls) / n
    var = sum((x - mean)**2 for x in ls) / n
    std_dev = var ** 0.5
    return std_dev
# create a list of data points
ls =df['Mthly_HH_Income']
standard_daviation = get_std_dev(ls)
print(standard_daviation)
```

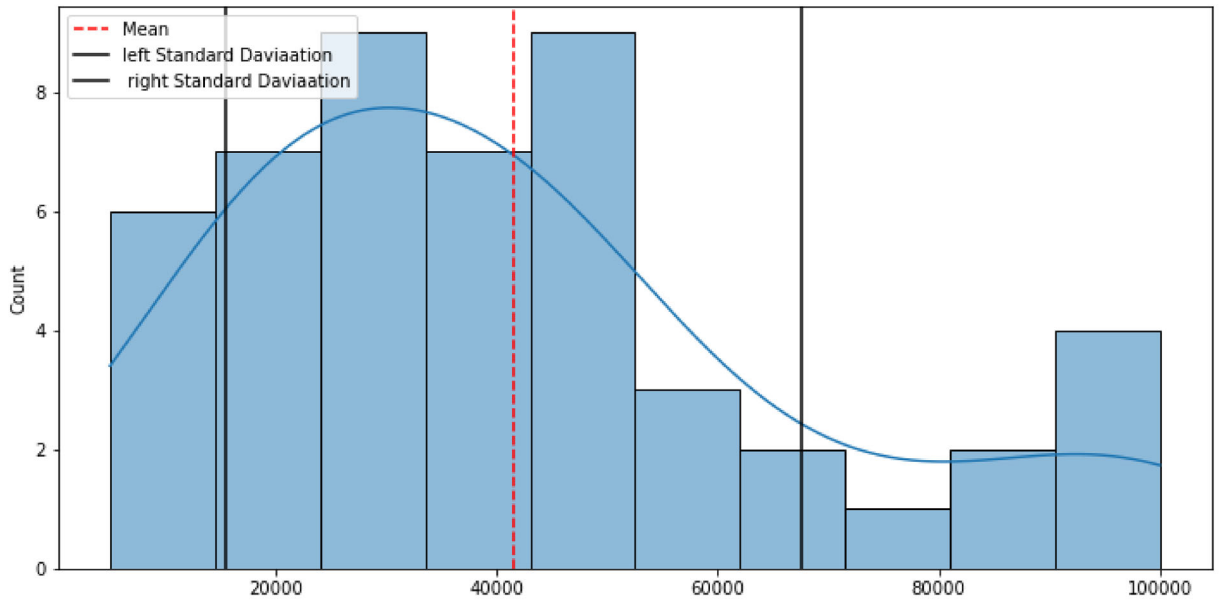
```
25835.611779092826
```

In [23]:
```
col =df['Mthly_HH_Income']
import numpy as np
def get_std_dev(col):
    N = len(col)
    mu = np.mean(col)
    sum_num = 0
    for j in range(len(col)):
        num = (col[j] - mu)**2
        sum_num = num +sum_num
    stdvariance = (sum_num/(N-1))**(0.5)
    return stdvariance
standard_daviation = get_std_dev(col)
print(standard_daviation)
```

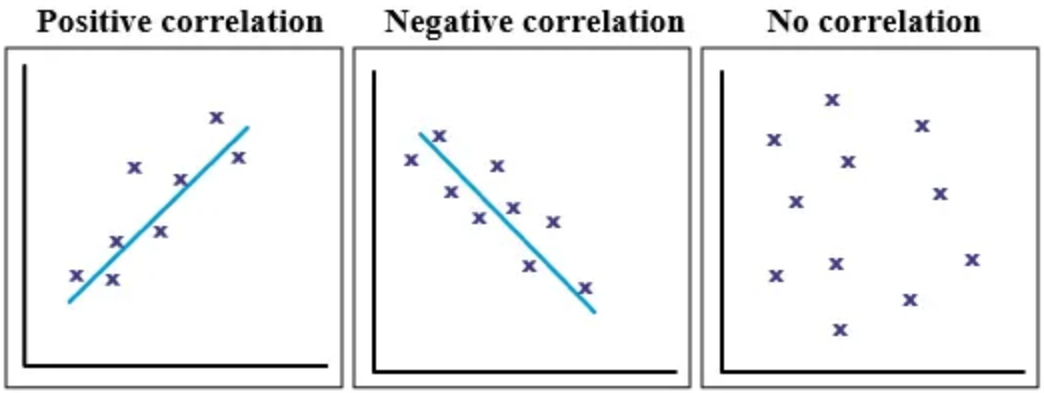
```
26097.908978713687
```

In [24]:
```
plt.figure(figsize =(12,6))
ax_hist =sns.histplot(list_nums, kde=True, bins=10 ,fill=20)
ax_hist.axvline(Mean, color='r', linestyle='--', label="Mean")
ax_hist.axvline((Mean-standard_daviation), color='black', linestyle='-', label="
ax_hist.axvline((Mean+standard_daviation), color='black', linestyle='-', label="

plt.legend(loc ='upper left')
```

Out[24]: <matplotlib.legend.Legend at 0x2ccc4c5a0a0>

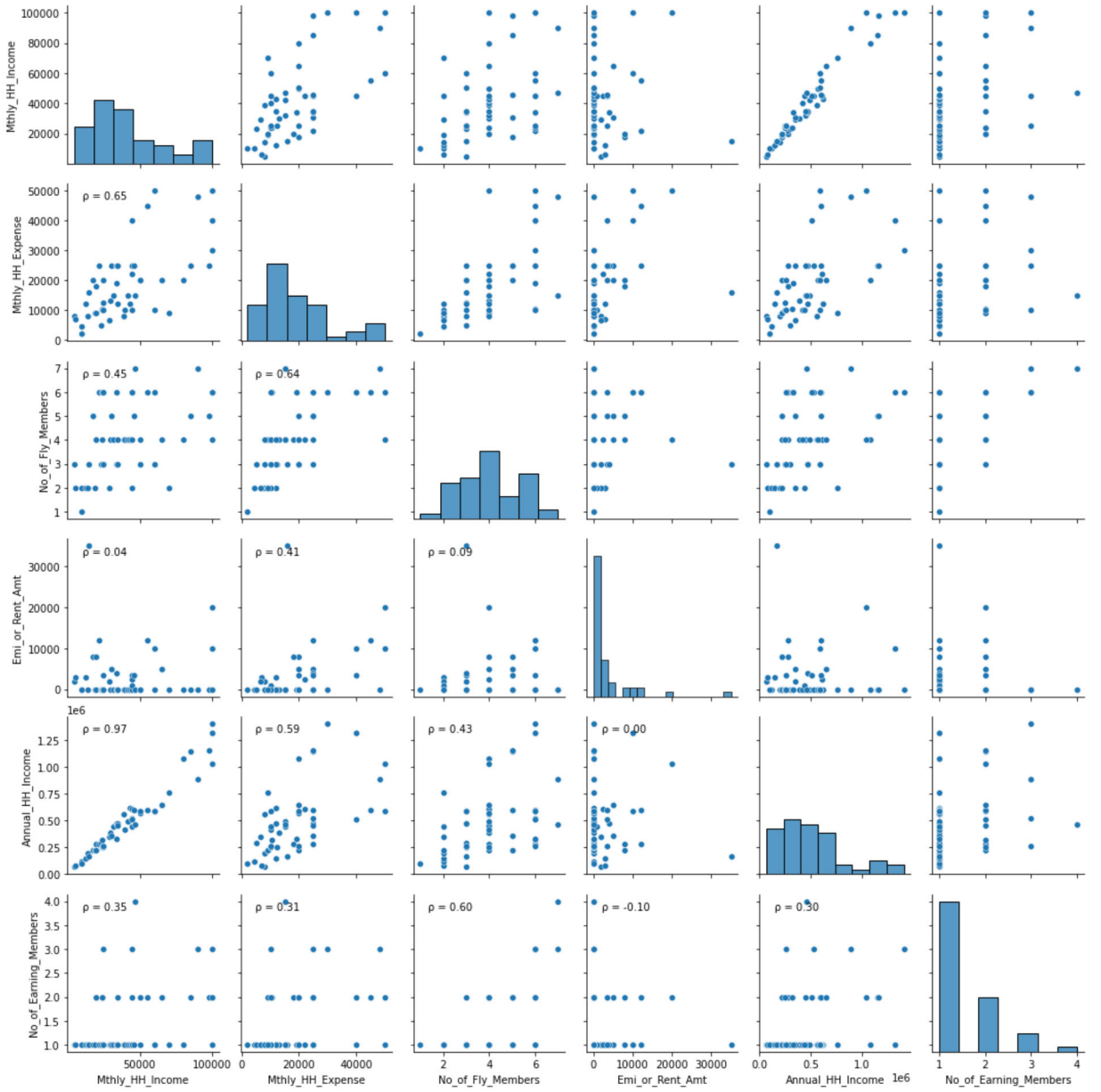


5. Correlation



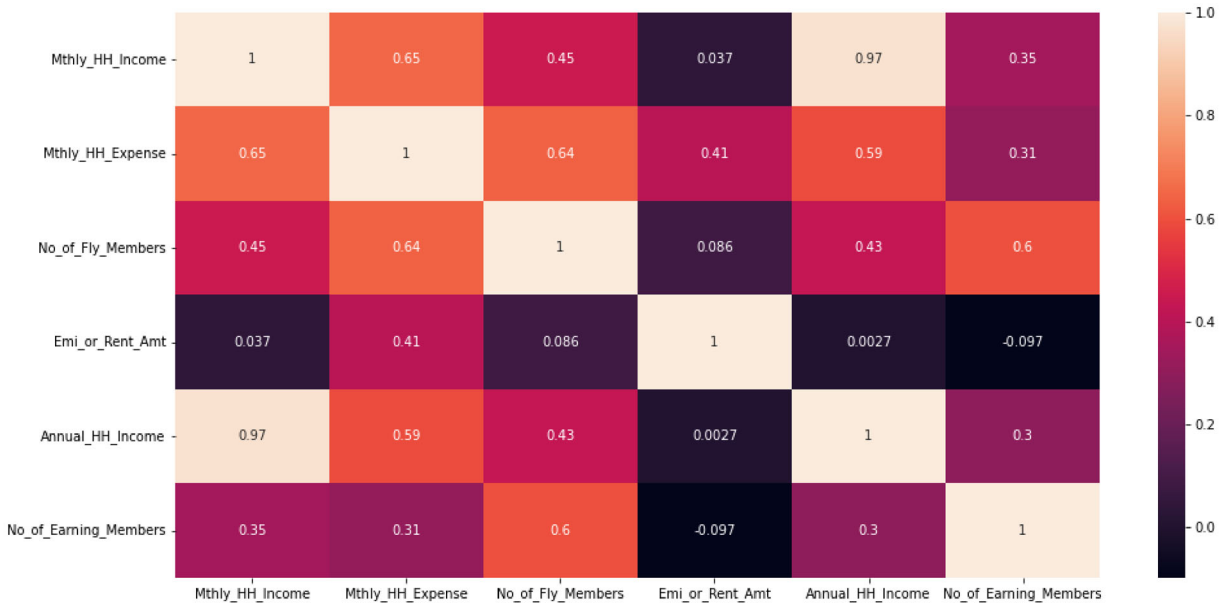
In [25]:
```
from scipy.stats import pearsonr
import matplotlib.pyplot as plt

def corrfunc(x, y, ax=None, **kws):
    """Plot the correlation coefficient in the top left hand corner of a plot."""
    r, _ = pearsonr(x, y)
    ax = ax or plt.gca()
    ax.annotate(f'ρ = {r:.2f}', xy=(.1, .9), xycoords=ax.transAxes)
```

In [26]:
```
g = sns.pairplot(df)
g.map_lower(corrfunc)
plt.show()
```



In [27]:
```
plt.figure(figsize=(16,8))
sns.heatmap(df.corr(), annot = True)
```

Out[27]: <AxesSubplot:>



- A correlation coefficient of 1 means that for every positive increase in one variable, there is a positive increase of a fixed proportion in the other. For example, shoe sizes go up in (almost) perfect correlation with foot length.
- A correlation coefficient of -1 means that for every positive increase in one variable, there is a negative decrease of a fixed proportion in the other. For example, the amount of gas in a tank decreases in (almost) perfect correlation with speed.
- Zero means that for every increase, there isn't a positive or negative increase. The two just aren't related.

Correlation - Pearson

The Pearson correlation coefficient is probably the most widely used measure for linear relationships between two normal distributed variables and thus often just called "correlation coefficient". Usually, the Pearson coefficient is obtained via a Least-Squares fit and a value of 1 represents a perfect positive relation-ship, -1 a perfect negative relationship, and 0 indicates the absence of a relationship between variables.

$$\rho = \frac{\operatorname{cov}(X, Y)}{\sigma_x \sigma_y}$$

And the estimate

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 \sum_{i=1}^{n}(y_i - \overline{y})^2}}$$

Taking Manual correlation

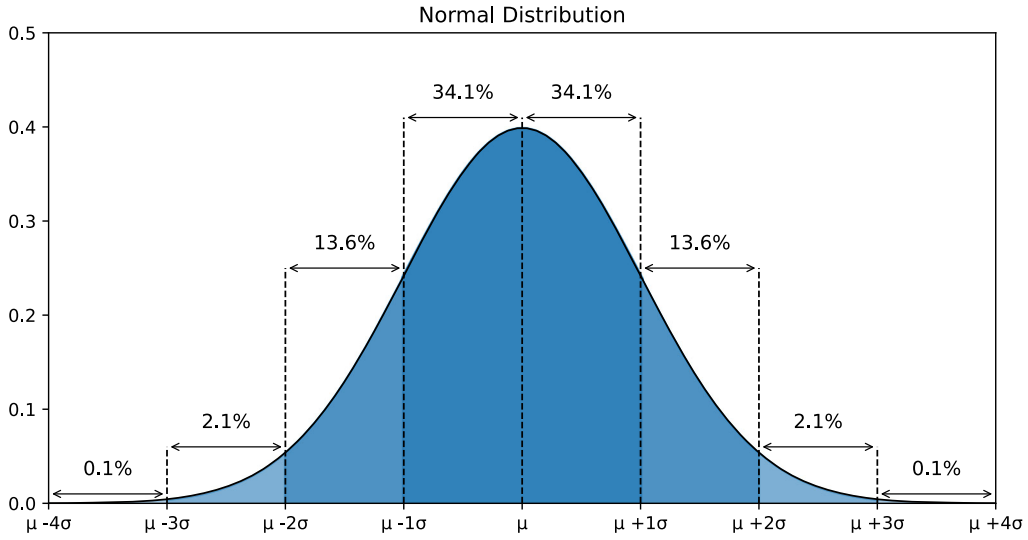
In [28]:
```
def corr(X,Y):
    import numpy as np
    sum_diff =0
    x_sum, y_sum = 0,0
    for i in range(len(X)):
        x_bar = np.mean(X)
        y_bar = np.mean(Y)
        diff = (X[i]-x_bar)*(Y[i]-y_bar)
        sum_diff = diff+sum_diff
        x_sum = ((X[i]-x_bar)**2)+x_sum
        y_sum = ((Y[i]-y_bar)**2)+y_sum
    correlation = sum_diff/(np.sqrt(x_sum*y_sum))
    return correlation
```

In [29]:
```
X = df.Emi_or_Rent_Amt
Y = df.No_of_Earning_Members
```

In [30]:
```
corr(X,Y)
```

Out[30]:
```
-0.09743128326031111
```

6. Normal Distribution



μ = mean

σ = Deviation

In above diagram colored portion indicate Area Under the Curve

For example (μ-2σ < Z < μ-1σ) this portion accupies the 13.6% area of total area under the curve

this example also shows the single interval

All normal distributions "look like" the one above.

- 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.

1. Normal distribution (univariate)

A random variable X is normally distributed with mean and variance if it has the probability density function of X as
Note: Single Variable

- Probability density function

$$\phi(\boldsymbol{x}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]$$

- This result is the usual bell-shaped curve that you see throughout statistics.
- squared difference between the variable x and its mean μ, $(x-\mu)^2$.

$\phi(\boldsymbol{x})$ value will be minimized when x is equal to μ. The quantity $-\left(\frac{x-\mu}{\sigma}\right)^2$ will take its largest value when x is equal to μ or likewise, since the exponential function is a monotone function, the normal density takes a maximum value when x is equal to μ.

The variance σ^2 defines the spread of the distribution about that maximum. If σ^2 is large, then the spread is going to be large, otherwise, if the σ^2 value is small, then the spread will be small.

As shorthand notation we may use the expression below:

$$\phi(\boldsymbol{x}) \sim N(\mu|\sigma^2)$$

2. Normal distribution (multivariate)

If we have a *d* x 1 random vector X that is distributed according to a multivariate normal distribution with population mean vector μ and population variance-covariance matrix Σ, then this random vector, X, will have the joint density function as shown in the expression below:

- Probability density function

$$\phi(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}\ |\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^t \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right]$$

Where,

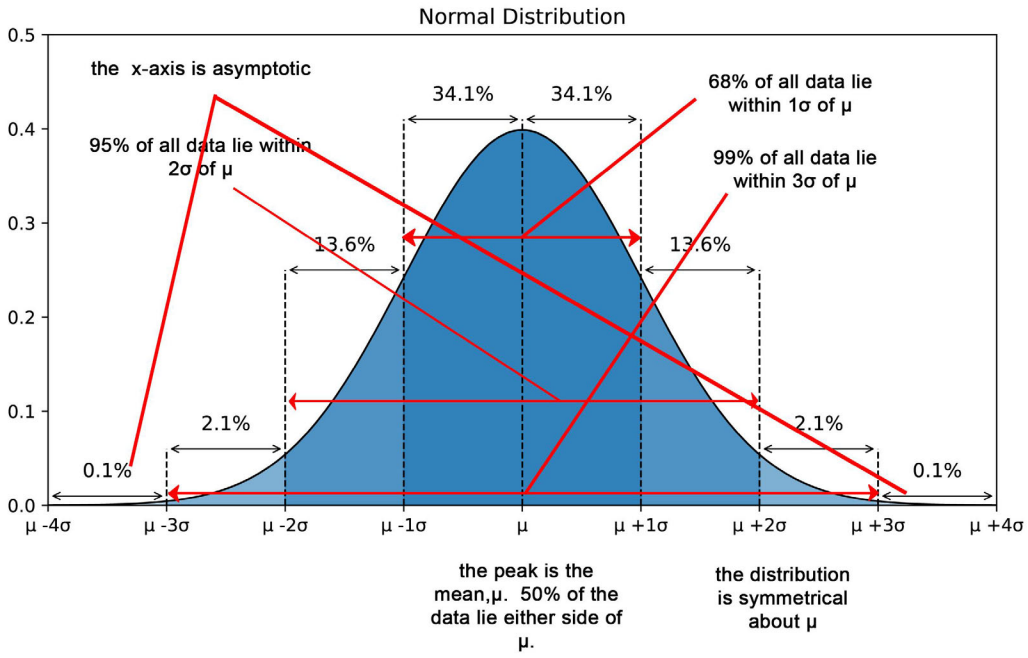
- $|\Sigma|$ determinant of the variance-covariance matrix Σ and Σ^{-1} is just the inverse of the variance-covariance matrix.
- this distribution will take maximum values when the vector X is equal to the mean vector μ, and decrease around that maximum

If *d* is equal to 2, then we have a bivariate normal distribution and this will yield a bell-shaped curve in three dimensions.

The shorthand notation, similar to the univariate version above, is

$$\phi(\boldsymbol{x}) \sim N(\boldsymbol{\mu}|\Sigma)$$

7. Feature of Normal Distribution

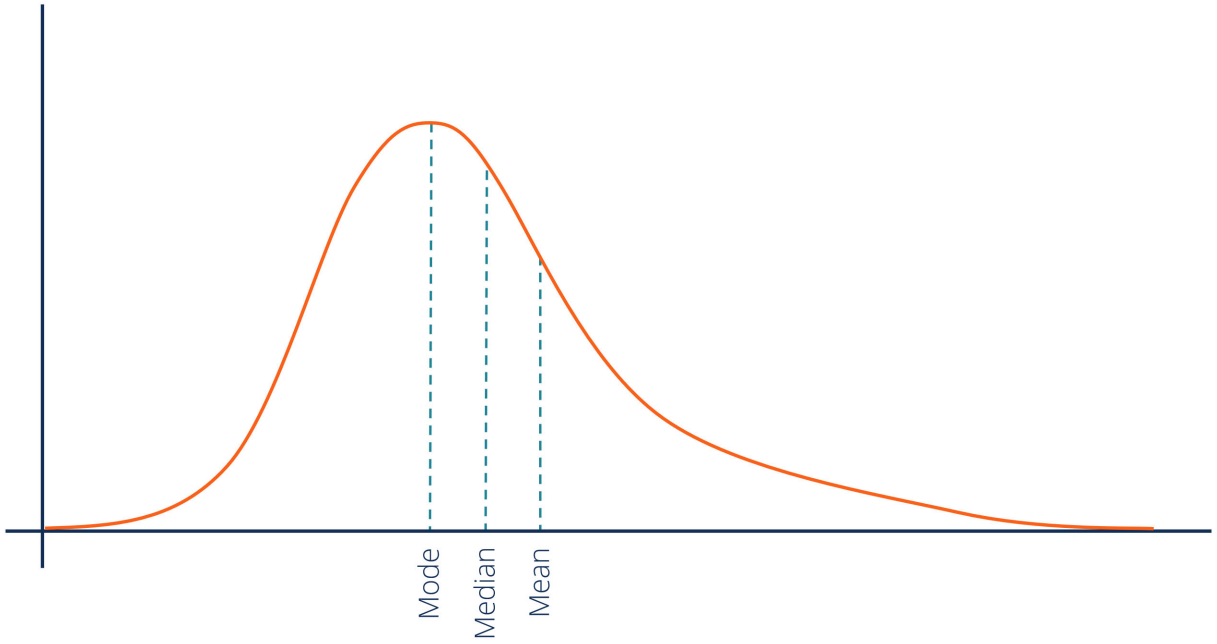


`

8. Positively Skewed & Negatively Skewed Normal Distribution

- **What is a Positively Skewed 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.



Central Tendency Measures in Positively Skewed Distributions

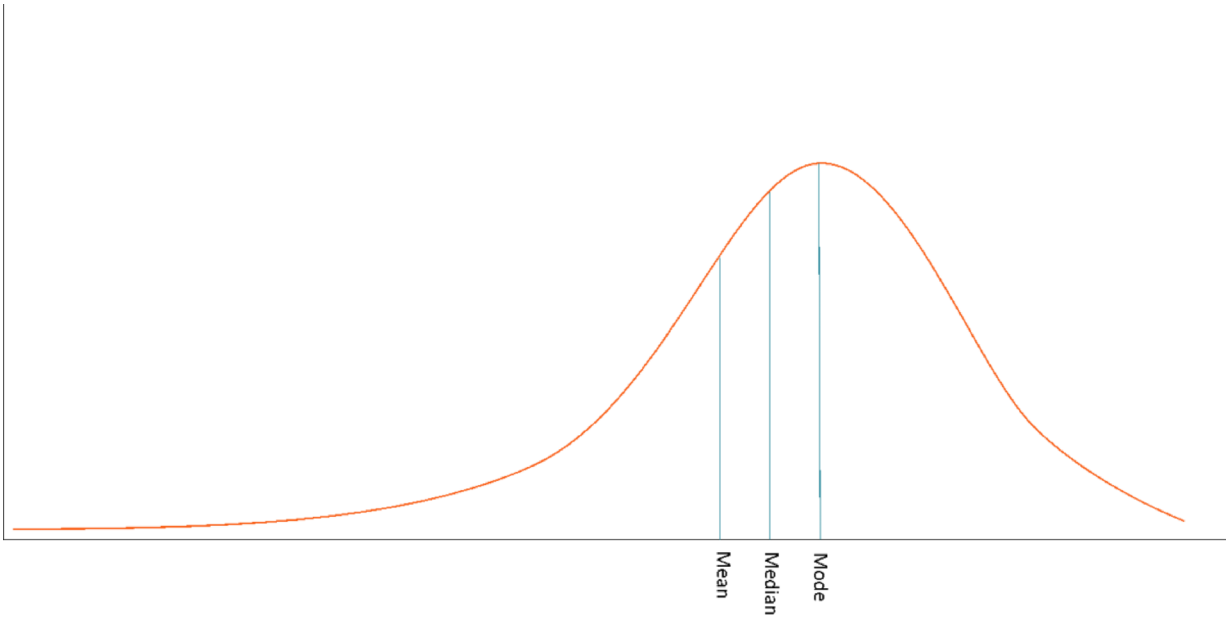
- normally distributed data where all measures of the central tendency (mean, median, and mode) equal each other, with positively skewed data, the measures are dispersed.

The general relationship among the central tendency measures in a positively skewed distribution may be expressed using the following inequality:

$$Mean > Median > Mode$$

- ## What is a Negatively Skewed 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.



While normal distribution is the most commonly encountered type of distribution, examples of the negatively skewed distributions are also widespread in real life. A negatively skewed distribution is the direct opposite of a positively skewed distribution.

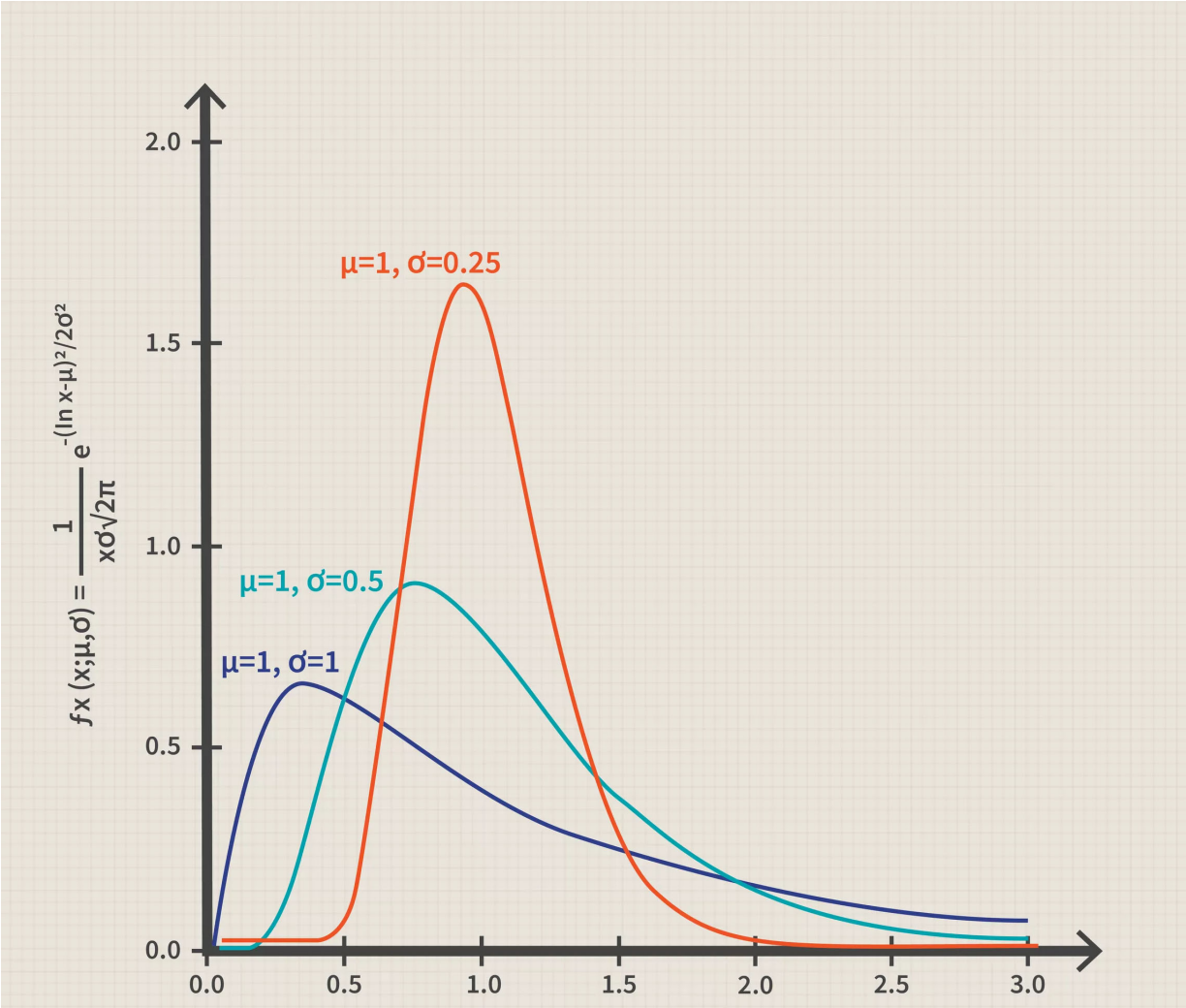
Central Tendency Measures in Negatively Skewed Distributions

- normally distributed data where all measures of central tendency (mean, median, and mode) equal each other, with negatively skewed data, the measures are dispersed.

The general relationship between the central tendency measures in a negatively skewed distribution may be expressed using the following inequality:

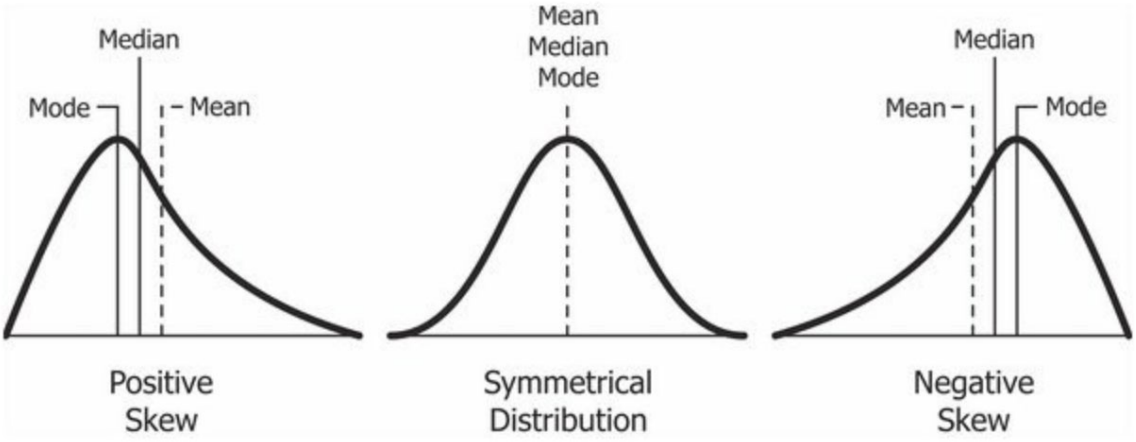
$$Mode > Median > Mean$$

The three probability distributions depicted below are **positively-skewed (or right-skewed)** to an increasing degree. **Negatively-skewed distributions are also known as left-skewed distributions.**



`

9. Effect on Mean, Median and Mode due to Skewness



`

10. Explain QQ Plot and show the implementation of the same

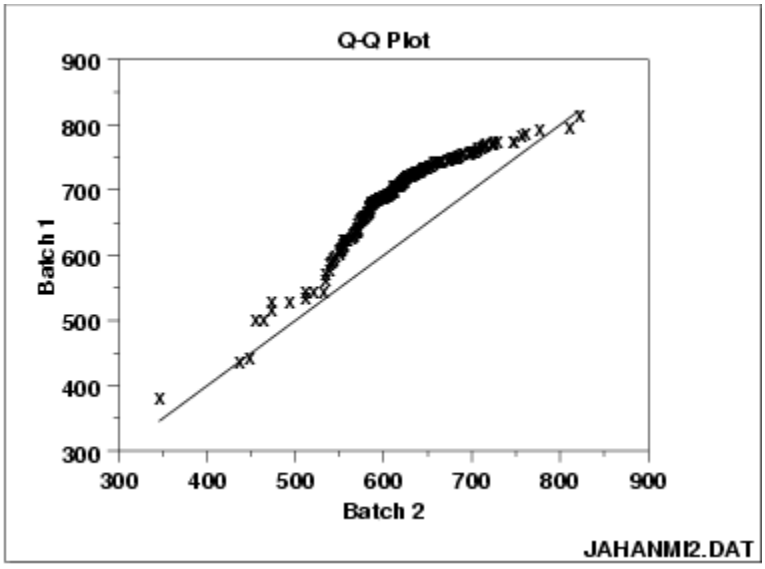
Quantile-Quantile Plot Source (https://www.itl.nist.gov/div898/handbook/eda/section3/qqplot.htm)

- The quantile-quantile (q-q) plot is a graphical technique for determining if two data sets come from populations with a common distribution.
- A q-q plot is a plot of the quantiles of the first data set against the quantiles of the second data set.
- By a quantile, we mean the fraction (or percent) of points below the given value. That is, the 0.3 (or 30%) quantile is the point at which 30% percent of the data fall below and 70% fall above that value.
- A 45-degree reference line is also plotted. If the two sets come from a population with the same distribution, the points should fall approximately along this reference line.
- The greater the departure from this reference line, the greater the evidence for the conclusion that the two data sets have come from populations with different distributions.

The advantages of the q-q plot are:

- The sample sizes do not need to be equal.
- Many distributional aspects can be simultaneously tested. For example, shifts in location, shifts in scale, changes in symmetry, and the presence of outliers can all be detected from this plot. For example, if the two data sets come from populations whose distributions differ only by a shift in location, the points should lie along a straight line that is displaced either up or down from the 45-degree reference line.

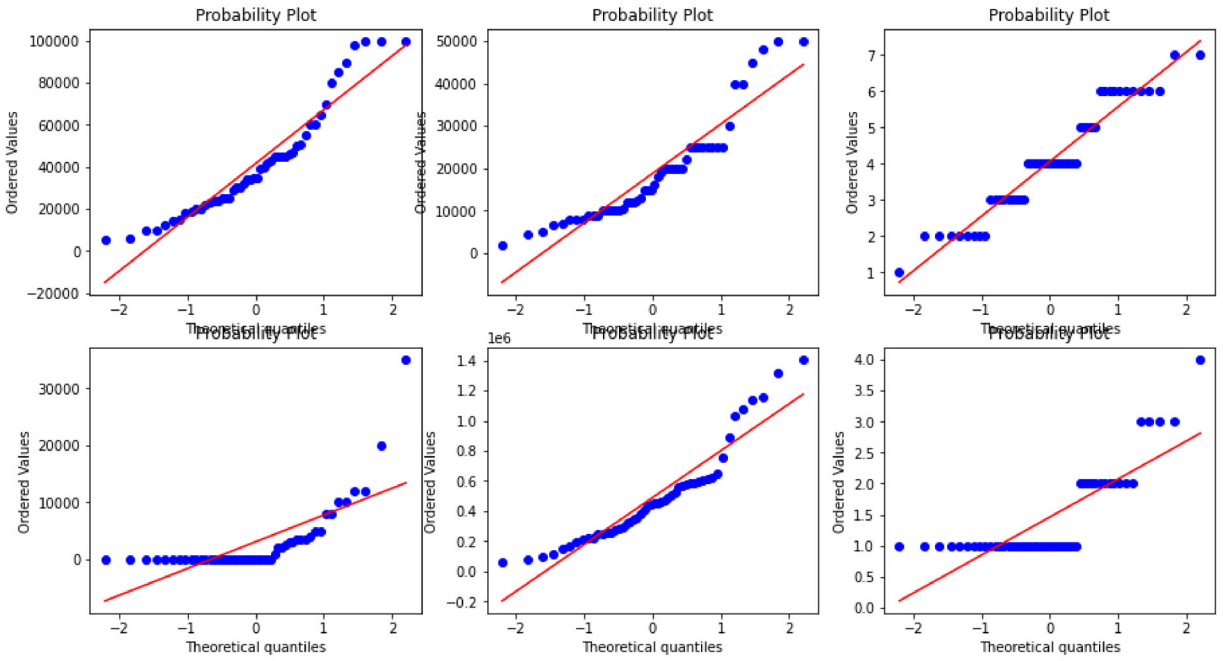
Sample Plot



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]:
```
from scipy import stats
plt.figure(figsize = (15,8))

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()
```



Conclusion: Columns are not normally distributed

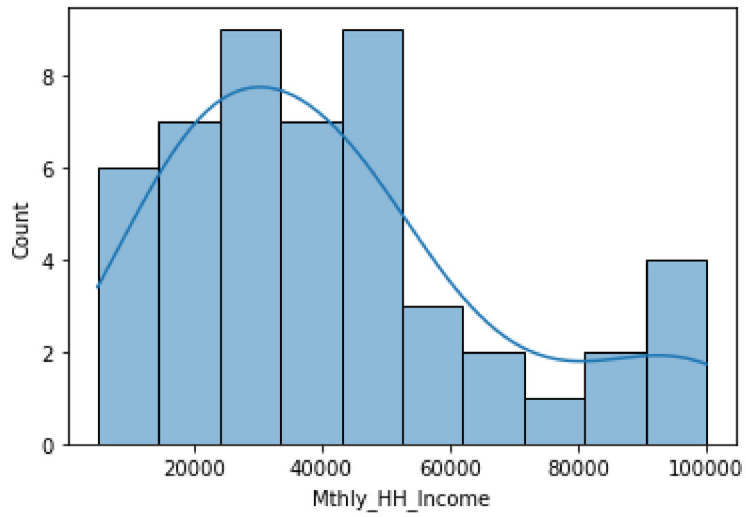
11. Explain Box Cox and show the implementation of the same

What is a Box Cox Transformation?

A Box Cox transformation is a transformation of non-normal dependent variables into a normal shape. Normality is an important assumption for many statistical techniques; if your data isn't normal, applying a Box-Cox means that you are able to run a broader number of tests.

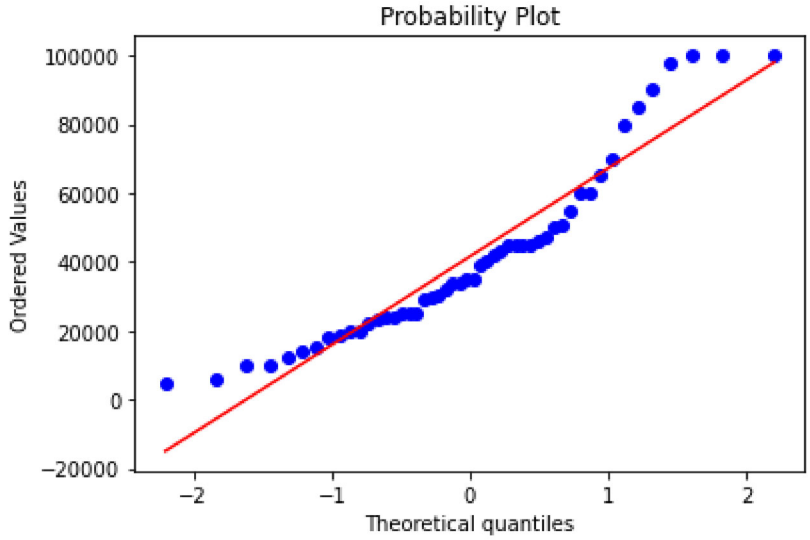
In [33]:
```
sns.histplot(df['Mthly_HH_Income'], kde=True, bins=10 ,fill=10)
```

Out[33]:
```
<AxesSubplot:xlabel='Mthly_HH_Income', ylabel='Count'>
```



ploting Q-Q plot to check the normality of the distribution

In [34]:
```
stats.probplot(df['Mthly_HH_Income'],dist="norm", plot = plt)
plt.show()
```

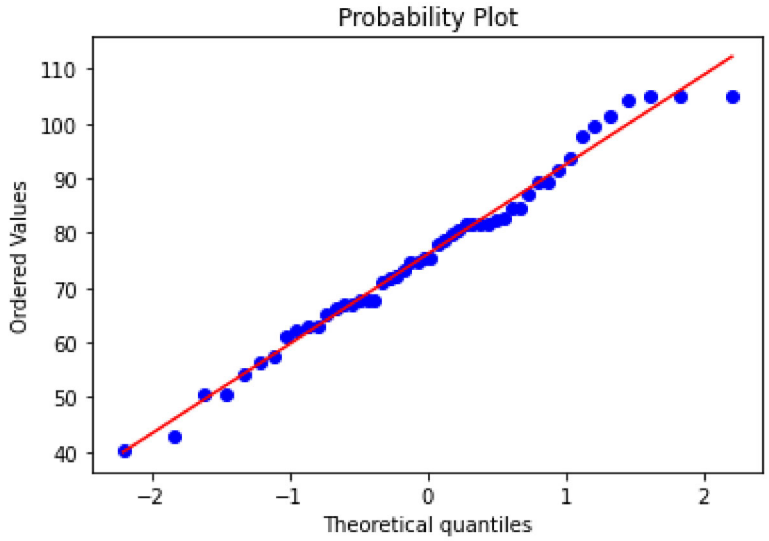


Box-cox transfomarion

In [35]:
```
pr_1, l = stats.boxcox(df['Mthly_HH_Income'])
print(pr_1, l)
```

```
[ 40.31831648  42.79683339  50.5170921   50.5170921   54.28331821
  56.29587859  57.55538753  61.01339013  62.0761407   63.10059552
  63.10059552  65.04699726  66.33720359  66.87370106  66.87370106
  67.74745911  67.74745911  67.74745911  71.01695067  71.78461975
  72.16178108  73.26802005  74.68814484  74.68814484  75.37645286
  75.37645286  78.0000836   78.62643549  79.84712626  80.44232498
  81.60438084  81.60438084  81.60438084  81.60438084  82.17194858
  82.73098204  84.35985154  84.62465737  86.92943718  89.3410009
  89.3410009   91.61630242  93.77269407  97.7826036   99.65741298
 101.45686676 104.19629334 104.85663515 104.85663515 104.85663515] 0.3031376789
702236
```

In [36]:
```
stats.probplot(pr_1, dist='norm', plot= plt)
plt.show()
```



In [37]:
```
sns.histplot(pr_1, kde=True, bins=10 ,fill=10)
plt.show()
```