Descriptive statistics

Measure & Analyze the Data (Structured Data)

Data

1. Structured Data

(i) Cross-Sectional Data: Data doesn't depend on time

- Continuous
- Discrete

(ii) Time Series Data: Data depends on time

2. Unstructured Data

- (i) Images & Videos
- (ii) Text & Audio

Types of Variables in Structured Data

Quantitative Data (Numerical Data)

- Data that is measured in numbers. It deals with numbers that make sense to a person for mathematical operations.

1. Continuous (values can be decimal)

- Refers to variables that can take on any numerical value
- Example: length, height, volume, age (22.3), stock price, etc.

2. Discrete Count (count data)

- Refers to variables that can only be measured in certain numbers
- Example: strength of a class, sales of mobiles, etc.

Qualitative Data (Discrete Categorical Data or Text Data)

- Refers to values that place things into different groups or categories

1. Nominal

- No natural ordering to the values of a categorical variable
- Example: species names, colors, brands

2. Ordinal

- Natural ordering to the values of a categorical variable
- Example: Grade A+ > Grade A > ..., ranks, (very likely, likely, ...)

`

Population

- Refers to set of all items or individuals of interest
- Ex: All voters in next election
- A Statistical Measure that describes the data from a population - **Parameter**
- Total Number of Population records = **N**

Sample

- A sample is a subset of population
- Ex: 1000 Voters selected
- A Statistical Measure that describes the data from a sample - **Statistic**
- Total Number of Sample records = **n**

Types of Statistics

- **Descriptive Statistics**
    - collecting, measuring, presenting and describing data
- **Inferential Statistics**
    - drawing conclusion and/or making decisions concerning a population b# Importing libraries

import numpy as np import pandas as pd## Measures of Central Tendency (1st Business Moment)

- **Mean, Median, Mode**
- Mean & Median are applied on continuous data
- Mode is applied on discrete data

Mean

- Sum of all data values / Total Number of Data values
- Mean- Population mean (μ) = (\frac{\sum X}{N})
- Sample mean (x̄) = (\frac{\sum x}{n})

m x}{n})

ased on sample data

In [70]:
```
import pandas as pd
# Creating a simple DataFrame
df = pd.DataFrame({"X": [1, 2, 3, 4, 5]})
df
```

Out[70]:

	X
0	1
1	2
2	3
3	4
4	5

In [71]:
```
# Calculating mean using built-in method
df["X"].mean()
# Calculating mean manually
df["X"].sum() / len(df)
```

Out[71]: 3.0

Median

- Refers to the data value that is positioned in the middle of an ordered dataset
- Median is applied on continuous data
- Median refers to the center value if you have an odd number of data points
- Median refers to the average of the center 2 values if you have an even number of data points

In [72]:
```
# New dataset to demonstrate median
df = pd.DataFrame({"X": [2, 4, 1, 9, 16, 10, 4, 8, 7]})
df
```

Out[72]:

	X
0	2
1	4
2	1
3	9
4	16
5	10
6	4
7	8
8	7

In [73]:
```python
df["X"].mean()
```

Out[73]: 6.777777777777778

In [74]:
```python
df = pd.DataFrame({"Y": [2, 4, 1, 9, 16, 10, 4, 8, 7, 5]})
df
```

Out[74]:

	Y
0	2
1	4
2	1
3	9
4	16
5	10
6	4
7	8
8	7
9	5

In [75]:
```python
df["Y"].mean()
```

Out[75]: 6.6

In [76]:
```python
df = pd.DataFrame({"Y": [2, 4, 1, 9, 16, 10, 4, 8, 7]})
df
```

Out[76]:

	Y
0	2
1	4
2	1
3	9
4	16
5	10
6	4
7	8
8	7

In [77]:
```python
df["Y"].median()
```

Out[77]: 7.0

Mode

- Most repeated value / Most frequent value
- **Unimodal Data** (if the data have only 1 mode value)
- **Bimodal Data** (if the data have 2 mode values)
- **Multimodal Data** (if the data have > 2 mode values)

In [78]:
```python
# DataFrame with multimodal, bimodal, and unimodal data
df = pd.DataFrame({
    "X": [1, 1, 2, 3, 4, 5],
    "Y": [1, 1, 2, 3, 3, 4],
    "Z": [1, 1, 2, 2, 3, 3],
    "I": [1, 2, 3, 4, 5, 6]
})
df
```

Out[78]:

	X	Y	Z	I
0	1	1	1	1
1	1	1	1	2
2	2	2	2	3
3	3	3	2	4
4	4	3	3	5
5	5	4	3	6

In [79]:
```
df["X"].mode()
```
Out[79]:
```
0    1
Name: X, dtype: int64
```
In [80]:
```
df["Y"].mode()
```
Out[80]:
```
0    1
1    3
Name: Y, dtype: int64
```
In [81]:
```
df["Z"].mode()
```
Out[81]:
```
0    1
1    2
2    3
Name: Z, dtype: int64
```
In [82]:
```
df["I"].mode()
```
Out[82]:
```
0    1
1    2
2    3
3    4
4    5
5    6
Name: I, dtype: int64
```

Measures of Dispersion or Measures of Spread(2nd business Moment)

Range,IQR,Variance,Std.deviation- all are applied only on continous variable only

In [83]:
```
# Sample data for dispersion measures
df = pd.DataFrame({"X": [1, 2, 3, 4, 5]})
df
```
Out[83]:

	X
0	1
1	2
2	3
3	4
4	5

Minimum

In [84]:
```
df["X"].min()
```
Out[84]:
```
1
```

Maximum

In [85]:
```
df["X"].max()
```
Out[85]:
```
5
```

Range

- Range = Maximum value - Minimum value

In [86]:
```
# Range calculation
df["X"].max() - df["X"].min()
```
Out[86]: 4

Deviation (X - μ)

- Deviation = data deviated from the mean = how dispersed the data is from the central value

In [87]:
```
df["X-μ"] = df["X"] - df["X"].mean()
df
```
Out[87]:

	X	X-μ
0	1	-2.0
1	2	-1.0
2	3	0.0
3	4	1.0
4	5	2.0

Mean Deviation

Mean Deviation = ∑((xi−μ)/N

- For any given data set, **mean deviation is always zero**

In [88]:
```
df["X-μ"].mean()
```
Out[88]: 0.0

Population Variance (σ^2)

σ2 = 1/N ∑(xi−μ)2

In [89]:
```
df["X"].var(ddof=0)
```
Out[89]: 2.0

Standard Deviation

- It is the **statistical measure of the dispersion of the dataset relative to its mean**
- It tells how close the values in the data set are to the mean
- High std deviation: values are largely deviated from the mean → spread is high
- Low std deviation: values are close to the mean

Population Standard Deviation (σ) = SQRT(1/N ∑(Xi−μ)2

In [90]:
```
df["X"].std(ddof=0)
```
Out[90]: 1.4142135623730951

Sample Variance s2= 1/(n-1) ∑(xi-x̄)

In [91]:
```
df["X"].var(ddof=1)
```
Out[91]: 2.5

Sample Std.dev = s= SQRT(1/(n-1) ∑(Xi−x̄)2

In [92]:
```
df["X"].std(ddof=1)
```
Out[92]: 1.5811388300841898

Coefficient of Variation

In [93]:
```
# Ratio of standard deviation to mean expressed in percentage
```

```
# CV = (σ / μ) * 100

import pandas as pd

df = pd.DataFrame({"X": [10,11,12,25,27,33,34,34,36,36,43,50,59]})

cv = df["X"].std(ddof=0) / df["X"].mean()
print("Coefficient of Variation (CV):", cv)
```

```
Coefficient of Variation (CV): 0.4493765924391731
```

Coefficient of variation

```
ratio of std. deviation and mean expressed in percentage = (σ / μ) * 100
It is the measure of variability of the dataset around its mean
The higher the CV the greater the std. deviation to its mean
```

In [94]:
```
import pandas as pd

df = pd.DataFrame({"X": [10,11,12,25,27,33,34,34,36,36,43,50,59]})

# Calculate Coefficient of Variation
cv = df["X"].std(ddof=0) / df["X"].mean()
cv
```

Out[94]: 0.4493765924391731

Percentile

Describes the percentage of data values that fall at or below the value # 0 percentile or Minimum : 0% of data is below this value # 25 percentile or Q1 : 25% of data is below this value # 50 percentile or Q2 : 50% of data is below this value # 75 percentile or Q3 : 75% of data is below this value # 100 percentile or Maximum : 100% of data is below this value

In [95]:
```
df = pd.DataFrame({"X":[10,11,12,25,27,31,33,34,36,36,43,50,59]})
df
```

Out[95]:

	X
0	10
1	11
2	12
3	25
4	27
5	31
6	33
7	34
8	36
9	36
10	43
11	50
12	59

In [96]:
```
# 0 percentile or Minimum
df["X"].quantile(0)
```

Out[96]: 10.0

In [97]:
```
# 25 percentile (Q1)
Q1 = df["X"].quantile(0.25)
Q1
```

Out[97]: 25.0

In [98]:
```
# 50 percentile (Q2)
Q2 = df["X"].quantile(0.5)
Q2
```

Out[98]: 33.0

In [99]:
```
# 75 percentile (Q3)
Q3 = df["X"].quantile(0.75)
Q3
```

Out[99]: 36.0

In [100…
```
# 100 percentile or Maximum
df["X"].quantile(1)
```

Out[100… 59.0

In [101…
```
# IQR = Q3 - Q1
IQR = Q3 - Q1
IQR
```

Out[101… 11.0

In [102…
```
# lower limit = Q1 - (1.5 * IQR)
ll = Q1 - (IQR * 1.5)
ll
```

Out[102… 8.5

In [103…
```
# upper limit = Q3 + (1.5 * IQR)
ul = Q3 + (IQR * 1.5)
ul
```

Out[103… 52.5

Outlier

- A data value that is numerically distant from a data set

What happens if outliers are available?

- Outliers will impact the statistical measures like mean, variance, standard deviation
- Outliers will more affect the mean, variance, standard deviation
- Outliers will less affect the median & IQR

How to calculate outliers?

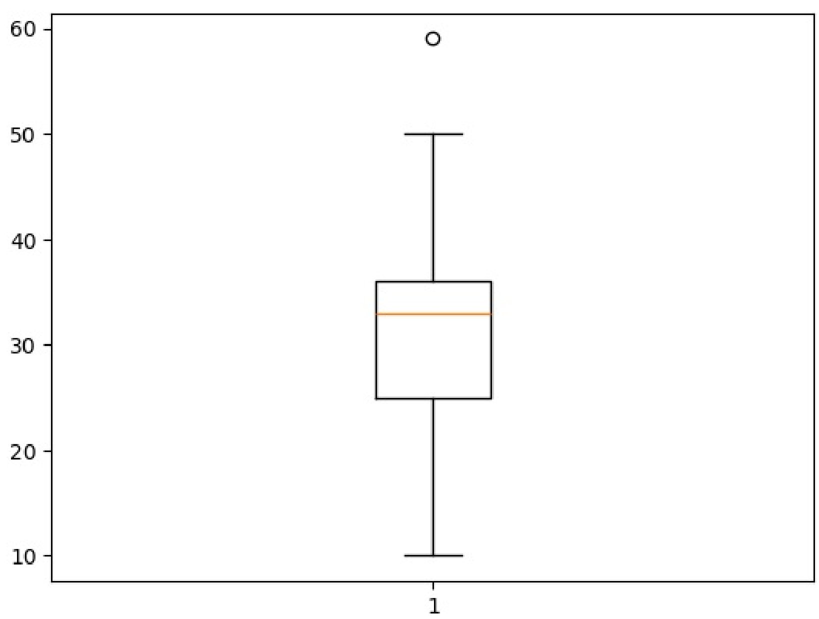
A data value is considered to be an outlier, if

```
datavalue < lower limit(Q1 - 1.5IQR)
datavalue > upper limit(Q3 + 1.5IQR)
```

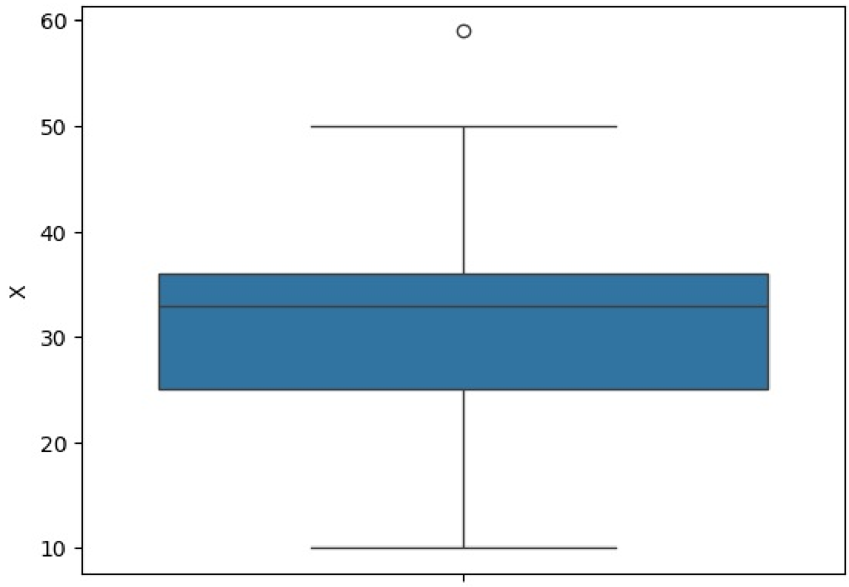
How to check outliers

- we use **box plot**

In [104…
```
import matplotlib.pyplot as plt
plt.boxplot(df["X"])
plt.show()
```



In [105…
```
import seaborn as sns
sns.boxplot(y=df["X"])
plt.show()
```



to extract outliers data

In [106…
```
df[(df["X"]<ll) | (df["X"]>ul)]
```

Out[106…

	X
12	59

In [107…
```
import pandas as pd

df = pd.DataFrame({
    "Gender": ["F","F","F","F","M","M","F","M","F","F","F","F","M","M","F","M"],
    "Marks": [30,41,42,51,52,53,61,62,68,69,77,78,79,88,89,100],
    "no_of_assignments": [1,1,1,1,2,2,2,3,3,3,3,3,4,4,4,4]
})

df
```

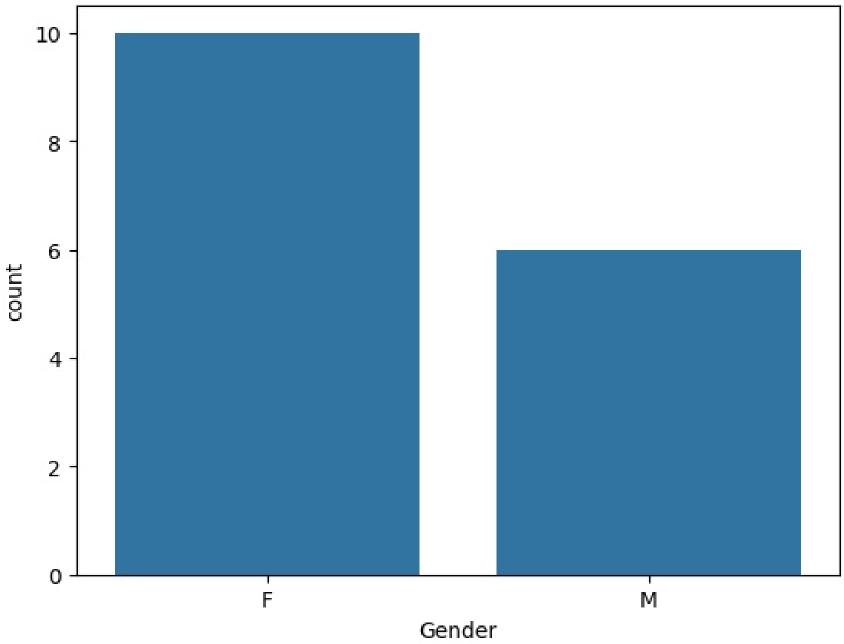
Out[107…

	Gender	Marks	no_of_assignments
0	F	30	1
1	F	41	1
2	F	42	1
3	F	51	1
4	M	52	2
5	M	53	2
6	F	61	2
7	M	62	3
8	F	68	3
9	F	69	3
10	F	77	3
11	F	78	3
12	M	79	4
13	M	88	4
14	F	89	4
15	M	100	4

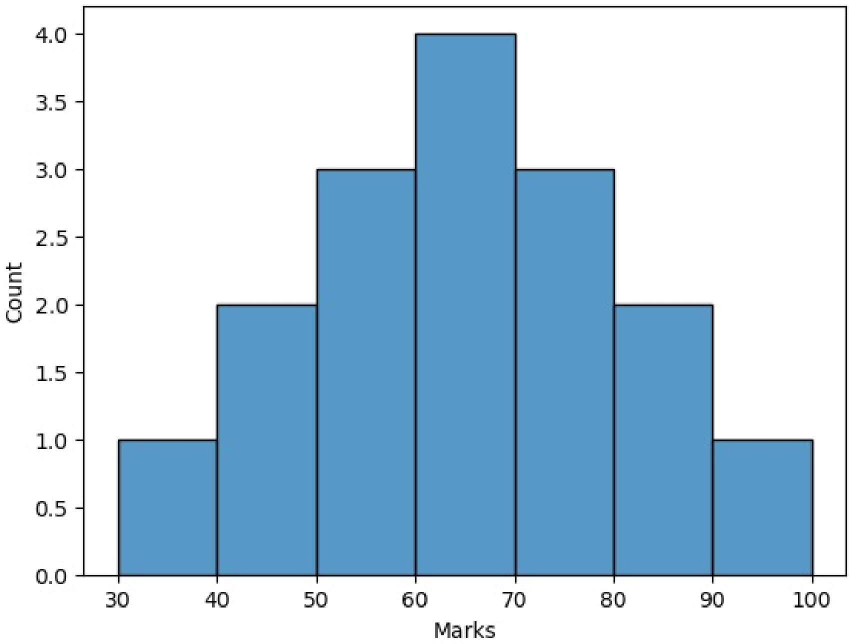
Frequency Distribution

- Graphical representation of variable with corresponding frequency.

Disrete Frequency Distribution : Graphical representation of discrete variable with corresponding frequency

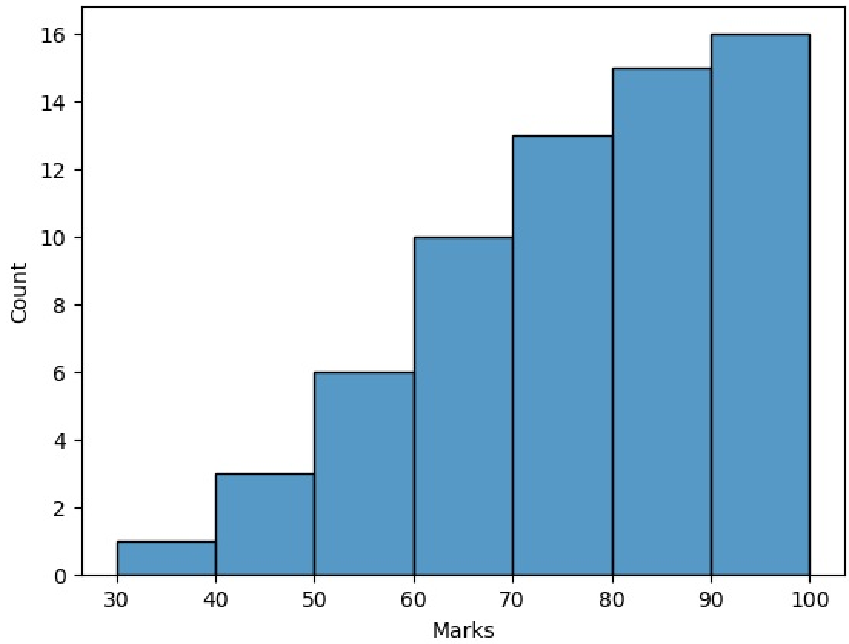
In [108…
```
df["Gender"].unique()
```
Out[108…
```
array(['F', 'M'], dtype=object)
```
In [109…
```
df["Gender"].value_counts()
```
Out[109…
```
Gender
F    10
M     6
Name: count, dtype: int64
```
In [110…
```
sns.countplot(x=df["Gender"])
plt.show()
```


Continous Frequency Distribution : Graphical representation of continous variable with corresponding frequency.

In [111…
```
sns.histplot(df['Marks'], bins=7, stat="count")
plt.show()
```


Cumulative Frequency Distribution

In [112…
```
sns.histplot(df["Marks"],bins=7,stat="count",cumulative=True)
plt.show()
```



Probability

- Chance of occurrence

Probability(requirement) = No. of values satisfies your requirementtotal no. of values

- **Example**: chance of occurrence of head when tossing a coin
- Always probability value lies between 0 to 1.
- Sum of all Probabilities = 1

In [113…
```
df["Gender"].value_counts() / len(df)
```

Out[113…
```
Gender
F    0.625
M    0.375
Name: count, dtype: float64
```

In [114…
```
df["Gender"].value_counts(normalize=True)
```

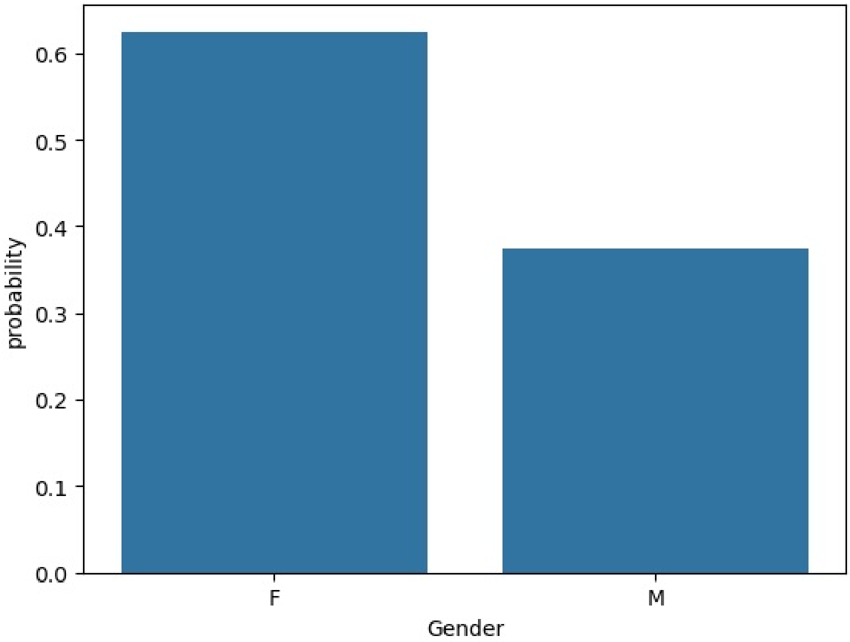
Out[114…
```
Gender
F    0.625
M    0.375
Name: proportion, dtype: float64
```

Probability Distribution

- Graphical representation of variable & respective probabilities of variable.

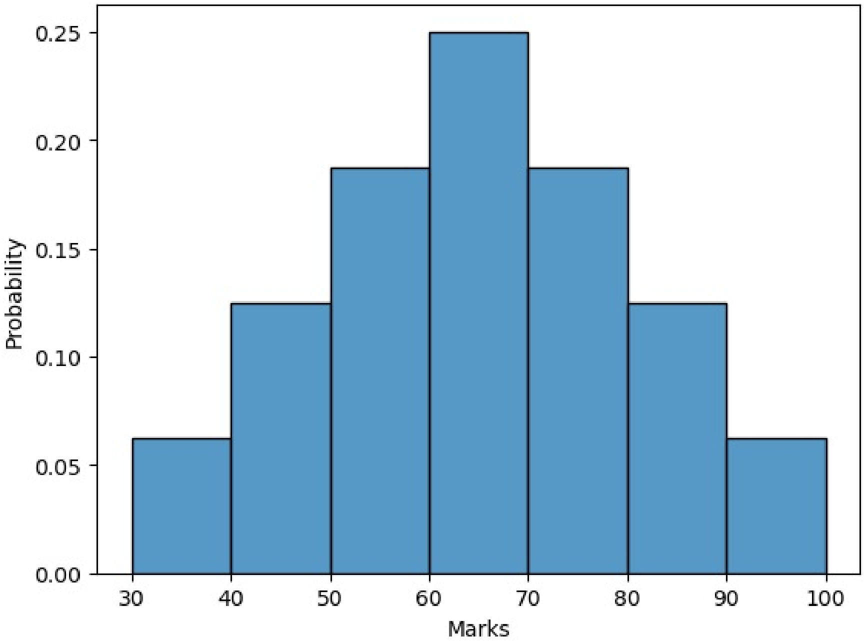
Discrete Probability Distribution : Graphical representation of discrete variable with corresponding probability

In [115…
```
sns.countplot(x=df["Gender"],stat="probability")
plt.show()
```



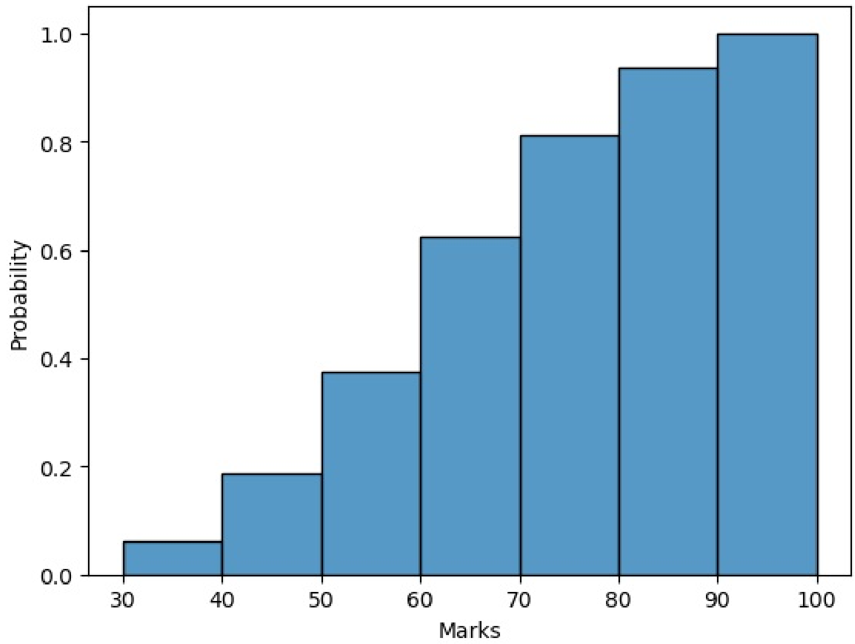
Continous Probability Distribution : Graphical representation of continous variable with corresponding probability

In [116…
```
sns.histplot(df["Marks"],bins=7,stat="probability")
plt.show()
```



Cumulative probability distribution

In [117…
```
sns.histplot(df["Marks"],bins=7,stat="probability",cumulative=True)
plt.show()
```



Measure of shape (3rd Business Moment)

skewness (s)

- it tells, whether the data is symmetrical or unsymmetrical distribution
- it describes asymmetry from the normal distribution
- Symmetrical Distribution is called as **Normal Distribution**
- Unsymmetrical Distribution is known as **Skewed Distribution**

Skewness =

$n(n-1)(n-2)\sum(x-\bar{x})^3 s^3$

In [118…
```
df = pd.DataFrame({"X":[1,2,3,4,5,6]})
df
```

Out[118…

	X
0	1
1	2
2	3
3	4
4	5
5	6

In [119…
```
df["X"].skew()
```

Out[119…
```
0.0
```

- If `Skewness = 0` then it is **Perfect Symmetrical or Perfect Normal Distribution**
- If `Skewness < 0` then it is said to be a **Negative Skewed or Left Skewed Distribution**
- If `Skewness > 0` then it is said to be a **Positive Skewed or Right Skewed Distribution**
- (-1 < Skewness < 1) then it is still considered to **Normal distribution**

In [120…
```
import pandas as pd

df = pd.DataFrame({
    "X": [0, 11, 12, 21, 22, 23, 31, 32, 38, 39, 47, 48, 49, 58, 59, 70],
    "Y": [0, 11, 12, 21, 22, 23, 24, 28, 29, 33, 34, 35, 37, 44, 59, 70],
    "Z": [0, 11, 12, 21, 22, 23, 34, 38, 49, 43, 44, 45, 47, 54, 59, 70]
})

df
```

Out[120…

	X	Y	Z
0	0	0	0
1	11	11	11
2	12	12	12
3	21	21	21
4	22	22	22
5	23	23	23
6	31	24	34
7	32	28	38
8	38	29	49
9	39	33	43
10	47	34	44
11	48	35	45
12	49	37	47
13	58	44	54
14	59	59	59
15	70	70	70

Symmetrical Distribution or Normal Distribution

Mean = Median

In [121…

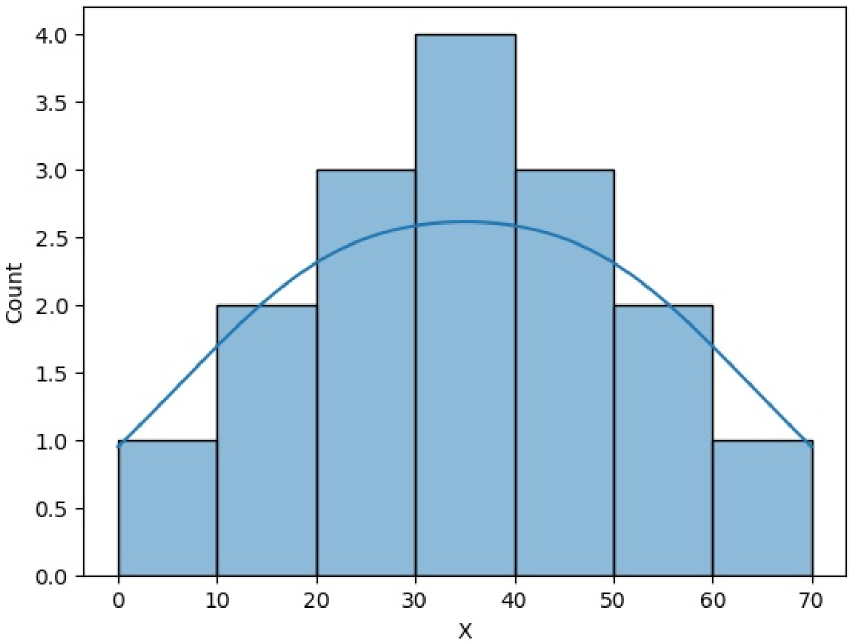
```
print("Mean of X:", df["X"].mean())
print("Median of X:", df["X"].median())
print("Skewness of X:", df["X"].skew())
```

```
Mean of X: 35.0
Median of X: 35.0
Skewness of X: 0.0
```

In [122…

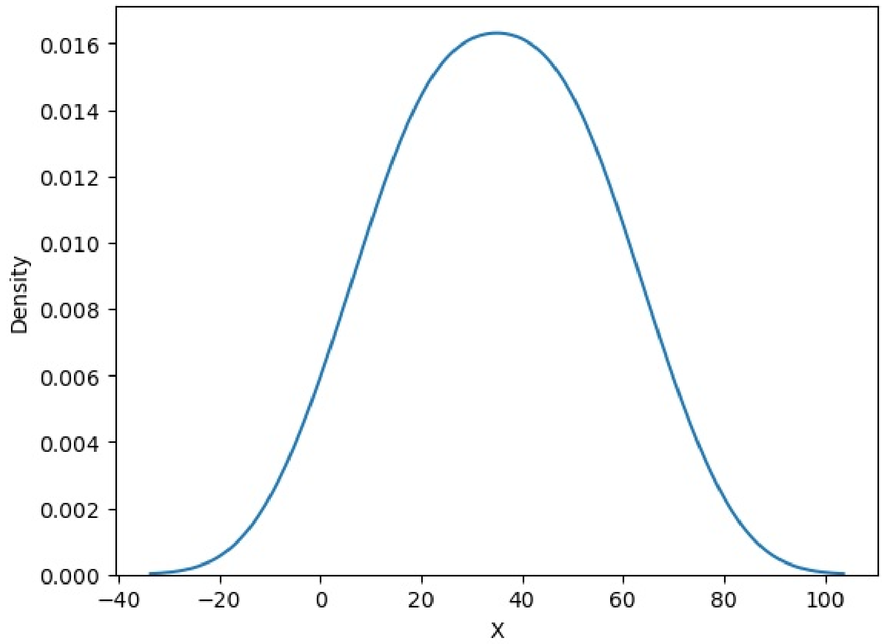
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.histplot(df["X"], bins=7, kde=True)
plt.show()
```



In [123…

```
sns.kdeplot(df["X"])
plt.show()
```



Right Skewed Distribution or **Positively Skewed Distribution**

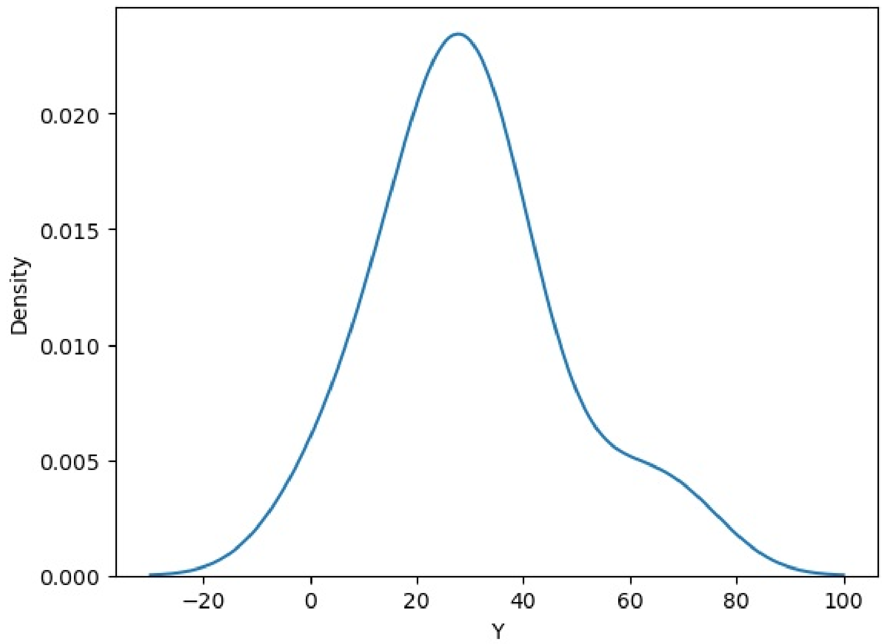
Mean > Median

In [124…

```
print("Mean of Y:", df["Y"].mean())
print("Median of Y:", df["Y"].median())
print("Skewness of Y:", df["Y"].skew())

sns.kdeplot(df["Y"])
plt.show()
```

```
Mean of Y: 30.125
Median of Y: 28.5
Skewness of Y: 0.6978985152470283
```



Left Skewed Distribution or **Negative Skewed Distribution**

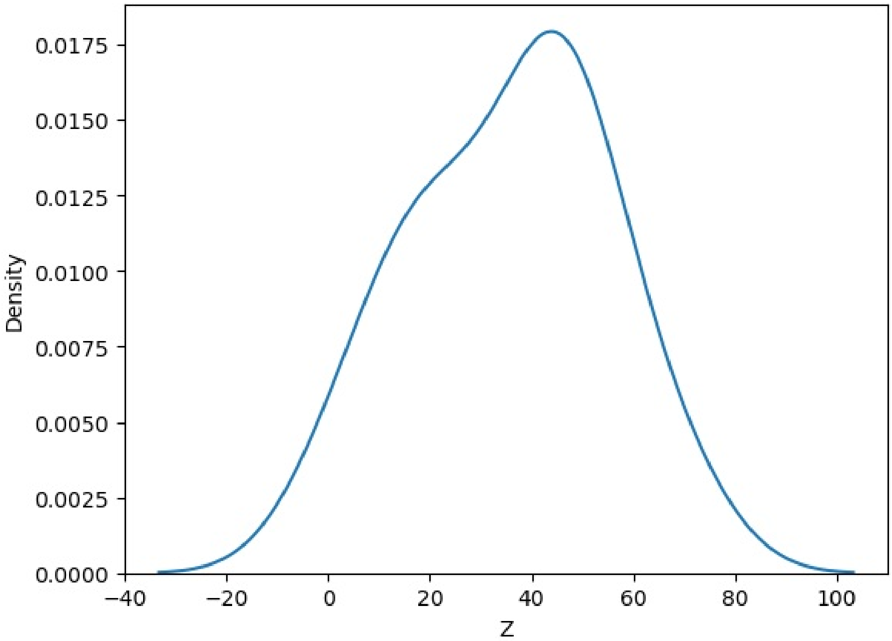
Mean < Median

In [125…

```
print("Mean of Z:", df["Z"].mean())
print("Median of Z:", df["Z"].median())
print("Skewness of Z:", df["Z"].skew())

sns.kdeplot(df["Z"])
plt.show()
```

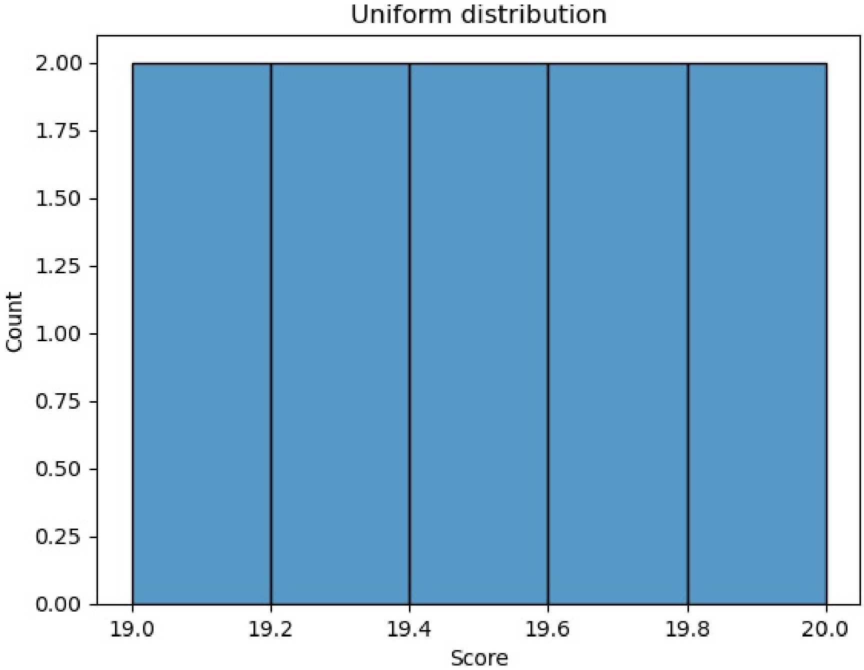
```
Mean of Z: 35.75
Median of Z: 40.5
Skewness of Z: -0.18882851815445098
```



In [126…

```
ds1 = pd.DataFrame({'Score':[19,19.1,19.2,19.3,19.4,19.5,19.6,19.7,19.8,20]})

sns.histplot(ds1['Score'])
plt.title('Uniform distribution')
plt.show()
```



In [ ]:

In [ ]:

For **Normal Distribution** Data, we have **68-95-99.7% Rule**

Question 1: The normal distribution data with mean of 70 & standard deviation of 10.
Approximately what area is contained between 70 and 90?

Question 2: Suppose that we gathered data from last mock test conducted at NareshIT and found that it followed Normal Distribution with mean of 60 & Standard Deviation of 10.
What proportion of students scored less than 49 in that exam?

Z-Score

$Z = x - \mu\sigma$

- **Standardization**: Converting all X values to corresponding Z-scores
- **Z-distribution**: Distribution of Z-scores is called Z-distribut.i-e**

Calculate probability using Z-score

In [127…
```python
# Z-score calculation for x = 49, mean = 60, std = 10
Zvalue = (49 - 60) / 10
Zvalue
```

Out[127… -1.1

In [128…
```python
from scipy import stats
stats.norm.cdf(Zvalue)
```

Out[128… 0.13566606094638267

Question3: When measuring the heights of all students at a local university, it was found that it was normally distributed with a mean height of 5.5 feet and standard deviation of 0.5 feet. What proportion of students are between 5.81 feet to 6.3 feet?

In [129…
```python
from scipy import stats

Z1 = (5.81 - 5.5) / 0.5
p1 = stats.norm.cdf(Z1)
print("probability of students less than 5.81:", p1)

Z2 = (6.3 - 5.5) / 0.5
p2 = stats.norm.cdf(Z2)
print("probability of students less than 6.3:", p2)

final = p2 - p1
print("probability of students between 5.81 to 6.3:", final)
```

```
probability of students less than 5.81: 0.7323711065310168
probability of students less than 6.3: 0.945200708300442
probability of students between 5.81 to 6.3: 0.21282960176942523
```

Central Limit Theorem:

- For continuous variable, Probability of a single value is **Zero**
- Since, Probability can't be calculated for a single value, we take an interval i.e., Point Estimate +- std. error

Standard Error = $\sigma\sqrt{n}$

- **The probability for the continuous variable is calculated on interval only for which CLT is used**

$$\left[\bar{x} - \sigma\sqrt{n},\ \bar{x} + \sigma\sqrt{n}\right]$$

Confidence Interval (CI):

$$ConfidenceInterval = \left[\bar{x} - (Z_{1-\alpha})\sigma\sqrt{n},\ \bar{x} + (Z_{1-\alpha})\sigma\sqrt{n}\right] ConfidenceInterval = \left[\bar{x} - (Z_{1-\alpha/2})\sigma\sqrt{n},\ \bar{x} + (Z_{1-\alpha/2})\sigma\sqrt{n}\right]$$

- α is the error
- If the confidence is 95%, then the error is 5%, so α = 0.05
- For 90% confidence: $Z_{1-\alpha}$ = 1.281 and $Z_{1-\alpha/2}$ = 1.645
- For 95% confidence: $Z_{1-\alpha}$ = 1.645 and $Z_{1-\alpha/2}$ = 1.96
- For 99% confidence: $Z_{1-\alpha}$ = 2.326 and $Z_{1-\alpha/2}$ = 2.576
- **As confidence increases, the interval range also increases**

In [130…
```python
from scipy import stats
```

```
stats.norm.cdf(-1.34)
```

Out[130… 0.09012267246445244

Calculate zscore using probability

In [131…
```
from scipy import stats
stats.norm.ppf(0.99)
```

Out[131… 2.3263478740408408

In [ ]:

Bivariate & Multivariate Analysis

In [132…
```
df = pd.DataFrame({
    "X": [11, 22, 13, 24, 30],
    "Y": [10, 9, 8, 7, 6],
    "Z": [18, 19, 21, 22, 40]
})
df
```

Out[132…

	X	Y	Z
0	11	10	18
1	22	9	19
2	13	8	21
3	24	7	22
4	30	6	40

Covariance:

- Covariance is used on two continuous variables, unlike variance (σ^2) which is a univariant
- It is represented as cov, for x and y variable it can be rep as cov(x,y)

$\text{cov}(x, y) = \sum_{ni=1}(x_i - \bar{x})(y_i - \bar{y})n - 1$

- in univariant variance since it is applied on a single variable that is var(x,x) which becomes $(x - \bar{x})(x - \bar{x})$
- The range of covariance values is $(-\infty, \infty)$
- Zero cov denotes no relation between two variables
- if cov is positive x and y are directly proportional, if it is negative then they are inversely proportional
- **In covariance sign is important not the value**
- it can be applied only on variables which have equal number of datapoints

In [133…
```
df.cov()
```

Out[133…

	X	Y	Z
X	62.50	-10.00	54.25
Y	-10.00	2.50	-11.75
Z	54.25	-11.75	82.50

Correlation:

- It measures the degree to which two variables are related to each other
- It is used to show how two variables are related, and is represented with **'r'**

$r = \text{cov}(x, y) S_x \cdot S_y = \sum(x - \bar{x})(y - \bar{y}) \sqrt{\sum(x - \bar{x})^2} \cdot \sqrt{\sum(y - \bar{y})^2}$

- **The values of 'r' lie within [-1, 1]**
- **In correlation, the value is important — not the sign**, based on the magnitude we can tell how related they are
- Higher the correlation value, the closer the data points, and the stronger the relationship

---

Understanding `corr()` Function:

- The `corr()` function is used to calculate the correlation between **two or more independent variables**. Correlation is a statistical measure that shows how strongly two variables are related to each other.

Interpretation of Correlation Coefficient |r|:

- (|r| = 1) ⟶ Perfect correlation
- (|r| > 0.8) ⟶ Strong correlation
- (0.5 \le |r| \le 0.8) ⟶ Moderate correlation
- (|r| < 0.5) ⟶ Weak correlation
- (|r| = 0) ⟶ No correlation

In [134…
```
df.corr()
```

Out[134…

	X	Y	Z
X	1.000000	-0.800000	0.755497
Y	-0.800000	1.000000	-0.818165
Z	0.755497	-0.818165	1.000000

In [ ]:

Processing math: 100%