Statistics

Definition of Statistics:

Statistics for Data Science: Statistics in the context of data science refers to the **collection, analysis, interpretation, presentation, and organization the data. It involves using mathematical and computational techniques to gather insights, make predictions, and draw conclusions from data.** Statistics plays a crucial role in various aspects of data science, including data exploration, hypothesis testing, pattern recognition, and machine learning model evaluation.

In data science, statistics provides methods for summarizing and describing data, identifying trends and patterns, quantifying uncertainty, and making informed decisions based on data-driven evidence. It encompasses concepts such as measures of central tendency (mean, median, mode), measures of dispersion (variance, standard deviation), probability distributions, hypothesis testing, regression analysis, and more.

In essence, statistics is the foundation that allows data scientists to extract meaningful information and knowledge from raw data, enabling them to derive insights and make informed decisions to solve complex problems across various domains.

To deal with data in statistics, the creators have divided the data into two different forms:

1. Descriptive statistics

2. Inferential statistics

Descriptive Statistics: Descriptive statistics are used to summarize and describe the main features of a dataset. They provide a simple overview of the data without making any inferences about the larger population. Some common descriptive statistics include:

1. **Measures of Central Tendency:**
    - Mean: The average of all values in a dataset.
    - Median: The middle value when the data is arranged in ascending order.
    - Mode: The value that appears most frequently in the dataset.
2. **Measures of Dispersion:**
    - Range: The difference between the maximum and minimum values in the dataset.
    - Variance: A measure of how much the values in the dataset vary from the mean.
    - Standard Deviation: The square root of the variance, indicating the average deviation from the mean.
3. **Frequency Distribution:**
    - A table that shows how frequently different values occur in the dataset.

Example of Descriptive Statistics: Consider the following dataset representing the ages of 10 individuals: 25, 30, 22, 40, 28, 35, 30, 22, 28, 32.

- Mean: (25 + 30 + 22 + 40 + 28 + 35 + 30 + 22 + 28 + 32) / 10 = 29.2
- Median: Since there are 10 values, the median is the average of the 5th and 6th values, which is (28 + 30) / 2 = 29.
- Mode: There are two modes in this dataset: 22 and 28, as they both appear twice.
- Range: 40 - 22 = 18
- Variance and Standard Deviation: Calculating these involves several steps, but they provide insights into the spread of data.

Inferential Statistics: Inferential statistics involve making inferences or predictions about a larger population based on a sample of data. This is often done through hypothesis testing and confidence intervals.

Example of Inferential Statistics: Let's say you're conducting a survey to estimate the average income of adults in a city. You collect data from a random sample of 200 adults and find that their average income is $50,000 with a standard deviation of 8,000. Using inferential statistics, you can construct a confidence interval (e.g., 95% confidence interval) to estimate the range within which the true average income of the entire population likely falls.

In this example, the average income of $50,000 represents your sample statistic, and the confidence interval gives you a range within which you believe the true population average income lies.

Remember that Descriptive statistics summarize data while inferential statistics help you make broader conclusions about populations based on sample data. Both types of

Difference between Descriptive Statistics and Inferential Statistics

Population Data and Sample Data:

In statistics, **a population refers to the entire group or set of individuals, items, or data points that share a common characteristic. This characteristic could be anything you're interested in studying, like the heights of all people in a country, the scores of all students in a school, or the sales data for a specific product across all stores.**

On the other hand, **a sample is a subset of the population that is selected for analysis. Since it's usually impractical or impossible to analyze an entire population, researchers often work with samples to draw conclusions about the larger population. The idea is that if the sample is selected properly and is representative of the population, the conclusions drawn from the sample can be generalized to the entire population.**

To ensure the validity of your conclusions, it's important to use proper sampling techniques to minimize bias and ensure that the sample accurately represents the population. Common sampling methods include:

- Random Sampling
- Stratified Sampling
- Cluster Sampling

For example, if you wanted to know the average height of people in a country, it would be quite difficult to measure the height of every single person. Instead, you could randomly select a representative sample of people from different regions and demographics within the country, measure their heights, and then use the average height of the sample to make an estimate about the average height of the entire population.

In summary, populations and samples are fundamental concepts in statistics. Populations are the entire group you're interested in, while samples are subsets of the population that are used to draw conclusions about the larger group. Proper sampling methods are essential to ensure that your findings from the sample are applicable to the entire population.

Sampling methods:

Sampling techniques are methods used in statistics to select a subset (sample) of individuals or items from a larger population for the purpose of making inferences about the whole population. Here are some common sampling techniques along with examples:

1. **Simple Random Sampling**: Each individual/item in the population has an **equal chance of being selected. This can be done using random number generators or drawing names from a hat.**

   Example: In a class of 100 students, a teacher assigns each student a number and then uses a random number generator to select 20 students for a survey.
2. **Stratified Sampling**: The population is **divided into distinct subgroups (strata) based on certain characteristics, and then random samples are taken from each stratum.**

   Example: In a town with a diverse population, a researcher divides the population into strata based on age groups (e.g., 0-18, 19-35, 36-60, 61+), and then selects a random sample from each age group.

3. **Systematic Sampling**: Every **nth individual/item is selected from the population. The starting point is selected randomly.**

   Example: In a factory with 1000 employees, every 10th employee on the employee list is selected for a satisfaction survey.
4. **Cluster Sampling**: The population is divided into clusters (groups), and a random sample of clusters is chosen. Then, all individuals/items within the selected clusters are included in the sample.

   Example: A country is divided into cities, and a random sample of cities is selected. All households within the selected cities are included in the survey.

Each sampling technique has its own strengths and weaknesses, and the choice of technique depends on the research objectives, available resources, and characteristics of the population being studied. It's important to select a technique that minimizes bias and accurately represents the population of interest.

In [4]:

```python
# Implementation of Sampling methods

# Random Sample

import random

# Function to perform simple random sampling
def simple_random_sampling(data, sample_size):
    return random.sample(data, sample_size)

# Collect user input for the population
population = []
population_size = int(input("Enter the population size: "))
for _ in range(population_size):
    value = input("Enter a value for the population: ")
    population.append(value)

# Collect user input for the desired sample size
sample_size = int(input("Enter the desired sample size: "))

# Perform simple random sampling
sample = simple_random_sampling(population, sample_size)

# Print the sample
print("Sample:", sample)

```

```
Enter the population size: 10
Enter a value for the population: 50
Enter a value for the population: 49
Enter a value for the population: 23
Enter a value for the population: 2
Enter a value for the population: 3
Enter a value for the population: 21
Enter a value for the population: 421
Enter a value for the population: 43
Enter a value for the population: 2143
Enter a value for the population: 2
Enter the desired sample size: 4
Sample: ['421', '43', '2143', '23']
```

In [7]:

```python
# Stratified sampling


import random

# Function to perform stratified sampling
def stratified_sampling(strata, sample_size_per_stratum):
    sample = []
    for values in strata.values():
        sample.extend(random.sample(values, sample_size_per_stratum))
    return sample

# Collect user input for strata and sample size
strata = {
    "Yes": [],
    "No": []
}

population_size = int(input("Enter the population size: "))
for _ in range(population_size):
    value = input("Enter a value for the population (Yes/No): ")
    if value in strata:
        strata[value].append(value)

sample_size_per_stratum = int(input("Enter the desired sample size per str

# Perform stratified sampling
sample = stratified_sampling(strata, sample_size_per_stratum)

# Print the sample
print("Sample:", sample)

```

```
Enter the population size: 6
Enter a value for the population (Yes/No): Yes
Enter a value for the population (Yes/No): No
Enter a value for the population (Yes/No): Yes
Enter a value for the population (Yes/No): Yes
Enter a value for the population (Yes/No): Yes
Enter a value for the population (Yes/No): No
Enter the desired sample size per stratum: 2
Sample: ['Yes', 'Yes', 'No', 'No']
```

In [9]:

```python
# Systematic Sampling


# Function to perform systematic sampling
def systematic_sampling(data, sample_size):
    n = len(data)
    step = n // sample_size
    start_index = 0  # You can also randomly select a starting index

    sample = []
    for i in range(sample_size):
        index = (start_index + i * step) % n
        sample.append(data[index])

    return sample

# Collect user input for the population
population = []
population_size = int(input("Enter the population size: "))
for _ in range(population_size):
    value = input("Enter a value for the population: ")
    population.append(value)

# Collect user input for the desired sample size
sample_size = int(input("Enter the desired sample size: "))

# Perform systematic sampling
sample = systematic_sampling(population, sample_size)

# Print the sample
print("Sample:", sample)

```

```
Enter the population size: 6
Enter a value for the population: A
Enter a value for the population: B
Enter a value for the population: C
Enter a value for the population: D
Enter a value for the population: E
Enter a value for the population: F
Enter the desired sample size: 2
Sample: ['A', 'D']
```

In [11]:

```python
# Clustering Sampling

import random

# Function to perform cluster sampling
def cluster_sampling(clusters, sample_size):
    selected_clusters = random.sample(clusters, sample_size)
    sample = []
    for cluster in selected_clusters:
        sample.extend(cluster)
    return sample

# Collect user input for the population
population = []
population_size = int(input("Enter the population size: "))
for _ in range(population_size):
    value = input("Enter a value for the population: ")
    population.append(value)

# Collect user input for the number of clusters and cluster size
num_clusters = int(input("Enter the number of clusters: "))
cluster_size = population_size // num_clusters

# Create clusters
clusters = [population[i:i + cluster_size] for i in range(0, population_s

# Collect user input for the desired sample size
sample_size = int(input("Enter the desired sample size: "))

# Perform cluster sampling
sample = cluster_sampling(clusters, sample_size)

# Print the sample
print("Sample:", sample)

```

```
Enter the population size: 10
Enter a value for the population: a
Enter a value for the population: b
Enter a value for the population: c
Enter a value for the population: d
Enter a value for the population: e
Enter a value for the population: a
Enter a value for the population: b
Enter a value for the population: c
Enter a value for the population: d
Enter a value for the population: y
Enter the number of clusters: 2
Enter the desired sample size: 2
Sample: ['a', 'b', 'c', 'd', 'e', 'a', 'b', 'c', 'd', 'y']
```

Types of Sampling Methods

Sampling Method	Description	Example
Random Sampling	• Gathering a representative sample from a population where each member in the population has an **equal chance** of being selected.	• Using a **random number generator** to select students in a class to complete a task.
Stratified Sampling	• Smaller groups or **strata** within the sample are represented **proportionally** to the population.	• Finding out a favourite soap opera from different age **categories** of people in a year group.
Systematic Sampling	• Every member in the population is given a **number**. After the first member is chosen at random, the remaining members are chosen from a given **interval**.	• A **list of people** with their first names in alphabetical order are numbered. The 5th person is **chosen randomly**, followed by every subsequent 8th person.
Non Random Sampling	• **Convenience** sampling is used for **ease** of data collection. **Volunteers** usually collect data.	• Asking people at a given location about how long their commute to work is.
Capture Recapture	• **Collecting** a sample of data from **one location** at **different points in time, marking** the individuals to estimate a population size.	• A sample of woodlice were **captured, marked and released**. Another sample of woodlice was captured 5 days later and the number of marked woodlice was counted.

Measure of Central Tendency:

Measures of central tendency are statistical measures that indicate the center or average of a dataset. They provide a single value that summarizes the location of the data points. The three most common measures of central tendency are the mean, median, and mode.

Here's an explanation of each measure along with a Python implementation example:

Mean: The mean, also known as the average, is calculated by summing up all the values in a dataset and dividing by the number of values.

In [12]:
```
def calculate_mean(data):
    return sum(data) / len(data)

dataset = [10, 20, 30, 40, 50]
mean = calculate_mean(dataset)
print("Mean:", mean)

```

```
Mean: 30.0
```

Median: The median is the middle value in a dataset when the values are arranged in ascending or descending order. If there's an even number of values, the median is the average of the two middle values.

In [13]:
```python
def calculate_median(data):
    sorted_data = sorted(data)
    n = len(sorted_data)
    if n % 2 == 0:
        middle_indices = [n // 2 - 1, n // 2]
        median = sum(sorted_data[i] for i in middle_indices) / 2
    else:
        median = sorted_data[n // 2]
    return median

dataset = [15, 30, 25, 40, 20]
median = calculate_median(dataset)
print("Median:", median)

```

```
Median: 25
```

Mode: The mode is the value that appears most frequently in a dataset. A dataset can have one mode, more than one mode (multimodal), or no mode (when all values occur with the same frequency).

In [14]:
```python
from collections import Counter

def calculate_mode(data):
    counter = Counter(data)
    modes = [value for value, count in counter.items() if count == max(co
    return modes

dataset = [10, 20, 30, 20, 40, 20, 50]
modes = calculate_mode(dataset)
print("Mode(s):", modes)

```

```
Mode(s): [20]
```

Remember that the choice of measure of central tendency depends on the nature of your data and what you want to convey. The mean is sensitive to outliers, while the median is more robust in the presence of outliers. The mode is useful for categorical data or when you want to identify the most frequent value.

$$\text{Mean} = \frac{\text{Sum of Observations}}{\text{Total Number of Observations}}$$

$$\text{If 'n' is odd:} \quad \text{Median} = \left(\frac{n+1}{2}\right)^{th} \text{term}$$

$$\text{If 'n' is even:} \quad \text{Median} = \frac{\left(\frac{n}{2}\right)^{th}\text{term} + \left(\frac{n}{2}+1\right)^{th}\text{term}}{2}$$

$$\text{Mode} = L + h\,\frac{(f_m - f_1)}{(f_m - f_1) + (f_m - f_2)}$$

Variables:

In statistics, a variable is a characteristic or attribute that can vary from one individual or item to another. Variables are the fundamental building blocks of data analysis, and they can be categorized into different types based on their nature and measurement scales. The two main types of variables are categorical and numerical (quantitative) variables.

Here's an explanation of each type along with examples:

1. **Categorical Variables**: Categorical variables represent qualitative characteristics with distinct categories or labels. They can be further divided into nominal and ordinal variables.
    - **Nominal Variables**: Nominal variables have categories with no inherent order or ranking.

      Example: Eye color (blue, brown, green), Gender (male, female, non-binary)
    - **Ordinal Variables**: Ordinal variables have categories with a specific order or ranking.

      Example: Education level (high school, college, graduate), Customer satisfaction rating (low, medium, high)
2. **Numerical Variables (Quantitative Variables)**: Numerical variables represent quantities and can be measured using numerical values. They can be further divided into discrete and continuous variables.
    - **Discrete Variables**: Discrete variables take on distinct, separate values.

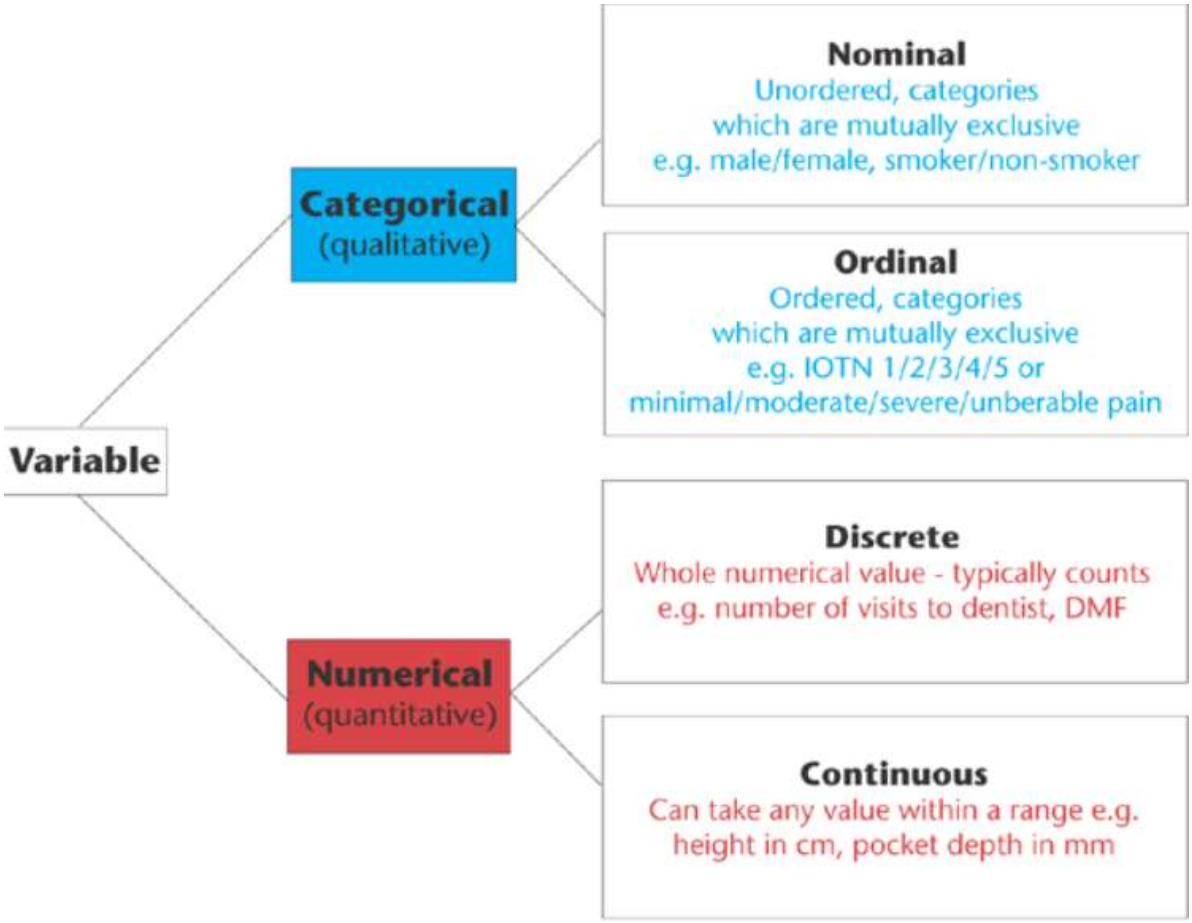
      Example: Number of children in a family, Number of items sold, Number of people in a room
    - **Continuous Variables**: Continuous variables can take on an infinite number of values within a range.

      Example: Height, Weight, Age, Temperature

Here are some examples of variables and their types:

1. **Variable**: Gender
   - **Type**: Categorical (Nominal)
   - **Possible Values**: Male, Female, Non-binary
2. **Variable**: Income
   - **Type**: Numerical (Continuous)
   - **Possible Values**: Any positive real number
3. **Variable**: Education Level
   - **Type**: Categorical (Ordinal)
   - **Possible Values**: High School, College, Graduate
4. **Variable**: Number of Pets
   - **Type**: Numerical (Discrete)
   - **Possible Values**: Non-negative integers (0, 1, 2, ...)
5. **Variable**: Temperature
   - **Type**: Numerical (Continuous)
   - **Possible Values**: Any real number within a range

Understanding the type of variables you're dealing with is crucial for selecting appropriate statistical techniques and making meaningful interpretations of your data. Different types of variables require different methods of analysis and visualization.



Dependent and Independent Variable

In statistics and research, the terms "dependent variable" and "independent variable" are used to describe the relationship between variables in an experiment or study. These terms are often used in the context of hypothesis testing, regression analysis, and other statistical analyses.

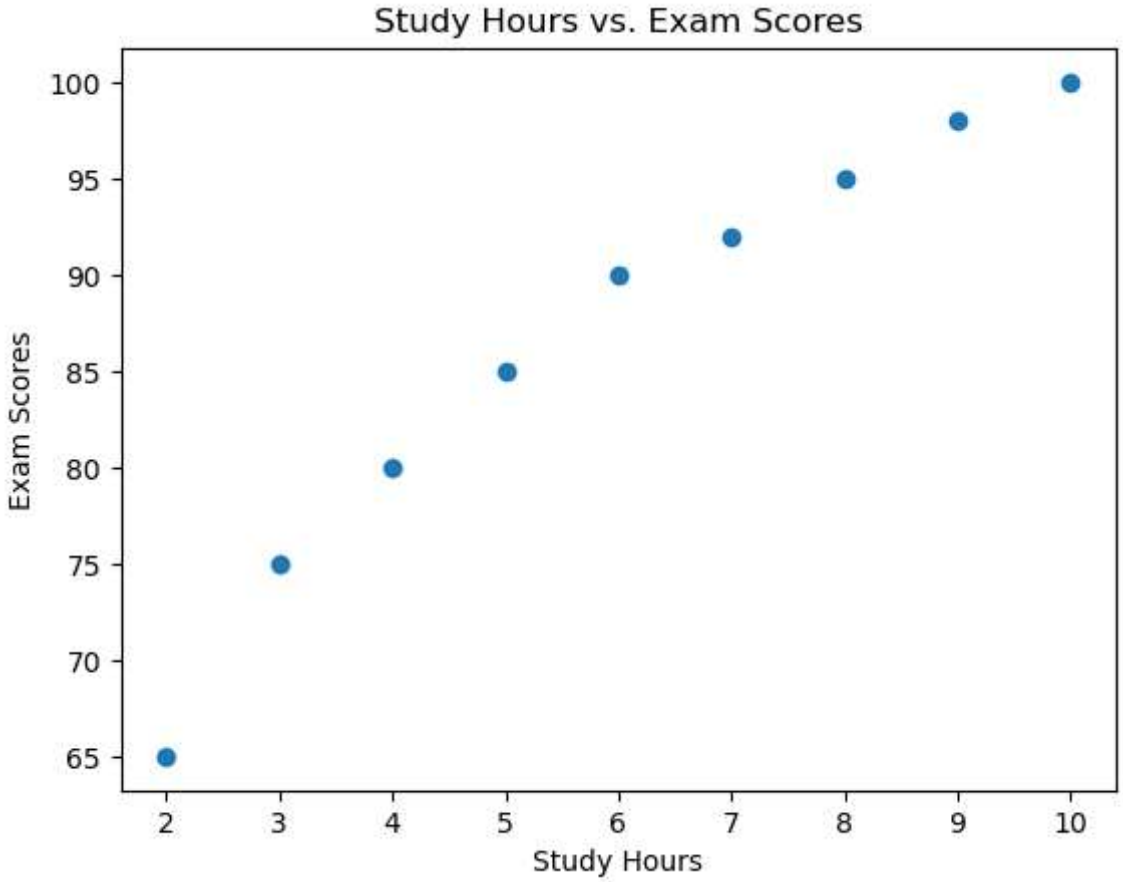
Independent Variable: The independent variable is the variable that is manipulated or controlled in an experiment. It is the cause or input that is believed to have an effect on the dependent variable.

Dependent Variable: The dependent variable is the variable that is observed or measured in response to changes in the independent variable. It is the outcome or result that is influenced by the independent variable.

Here's an explanation along with a Python example:

Let's consider an example where we want to study the effect of study hours on exam scores.

In [15]:
```
import matplotlib.pyplot as plt
import numpy as np

# Simulated data
study_hours = np.array([2, 3, 4, 5, 6, 7, 8, 9, 10])
exam_scores = np.array([65, 75, 80, 85, 90, 92, 95, 98, 100])

# Plot the data
plt.scatter(study_hours, exam_scores)
plt.xlabel("Study Hours")
plt.ylabel("Exam Scores")
plt.title("Study Hours vs. Exam Scores")
plt.show()

```



In this example, the independent variable is "Study Hours," as it is the variable we are manipulating (or selecting values for). The dependent variable is "Exam Scores," as it is the variable we are observing and measuring in response to changes in the independent variable.

The scatter plot illustrates the relationship between the two variables. As study hours increase, we can observe how exam scores tend to increase, indicating a potential positive relationship between study hours and exam scores.

In statistical analysis, you might use techniques like linear regression to quantify the relationship between the independent and dependent variables, and to make predictions based on this relationship.

Measure of Dispersion

Measures of dispersion, also known as measures of variability or spread, quantify the extent to which data points in a dataset are spread out or deviate from the central tendency (mean, median, or mode). These measures provide important insights into the distribution and variability of the data. Common measures of dispersion include the range, variance, standard deviation, and interquartile range.

Here's an explanation of each measure along with their formulas and a brief Python example:

Range: The range is the simplest measure of dispersion and is calculated as the difference between the maximum and minimum values in the dataset.

Formula: ***Range = Maximum Value - Minimum Value***

In [17]:
```
data = [12, 15, 18, 20, 23, 25, 30, 35, 40]
data_range = max(data) - min(data)
print("Range:", data_range)

```

```
Range: 28
```

Variance: Variance measures how much the values in a dataset vary from the mean. It's the average of the squared differences between each data point and the mean.

Formula: **Variance = Σ((xi - mean)^2) / (n - 1)**

Population Variance	Sample Variance
$\sigma^2 = \frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N}$	$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$
σ^2 = population variance	s^2 = sample variance
x_i = value of i^{th} element	x_i = value of i^{th} element
μ = population mean	$\bar{x}$ = sample mean
N = population size	n = sample size

In [19]:
```
import numpy as np

data = [12, 15, 18, 20, 23, 25, 30, 35, 40]
data_mean = np.mean(data) # mean value
squared_deviations = [(x - data_mean) ** 2 for x in data]
variance = sum(squared_deviations) / (len(data) - 1)
print("Variance:", variance.round(2))

```

```
Variance: 86.44
```

Standard Deviation: The standard deviation is the square root of the variance. It measures the average distance between each data point and the mean.

Formula: **Standard Deviation = √Variance**

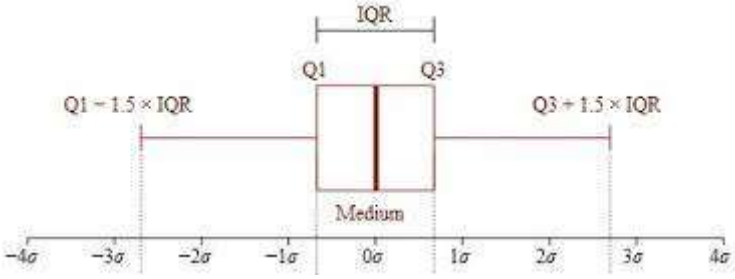
Standard Deviation Formula	
Population	**Sample**
$\sigma = \sqrt{\frac{\sum(X-\mu)^2}{N}}$	$s = \sqrt{\frac{\sum(X-\bar{x})^2}{n-1}}$
X – The Value in the data distribution μ – The population Mean N – Total Number of Observations	X – The Value in the data distribution $\bar{x}$ – The Sample Mean n - Total Number of Observations

In [21]:
```
standard_deviation = np.sqrt(variance)
print("Standard Deviation:", standard_deviation.round(2))

```

```
Standard Deviation: 9.3
```

Interquartile Range (IQR): The IQR is a measure of the spread of the middle 50% of the data. It's the difference between the third quartile (Q3) and the first quartile (Q1).

Formula: **IQR = Q3 - Q1**



In [22]:
```
q1 = np.percentile(data, 25)
q3 = np.percentile(data, 75)
iqr = q3 - q1
print("Interquartile Range:", iqr)

```

```
Interquartile Range: 12.0
```

These measures of dispersion provide valuable insights into the spread and variability of data. Different measures might be more appropriate in different contexts, so it's important to choose the one that suits the characteristics of your dataset and the goals of your analysis.

Empirical Rule

The Empirical Rule, also known as the 68-95-99.7 rule or the Three-Sigma Rule, is a rule of thumb that provides a rough guideline for understanding the distribution of data in a normal (Gaussian) distribution. This rule states that for a normal distribution:

- Approximately 68% of the data falls within one standard deviation of the mean.
- Approximately 95% of the data falls within two standard deviations of the mean.
- Approximately 99.7% of the data falls within three standard deviations of the mean.

In mathematical terms:

- About 68% of data falls within the interval [μ - σ, μ + σ].
- About 95% of data falls within the interval [μ - 2σ, μ + 2σ].
- About 99.7% of data falls within the interval [μ - 3σ, μ + 3σ].

Where:

μ is the mean (average) of the distribution.

σ is the standard deviation of the distribution.

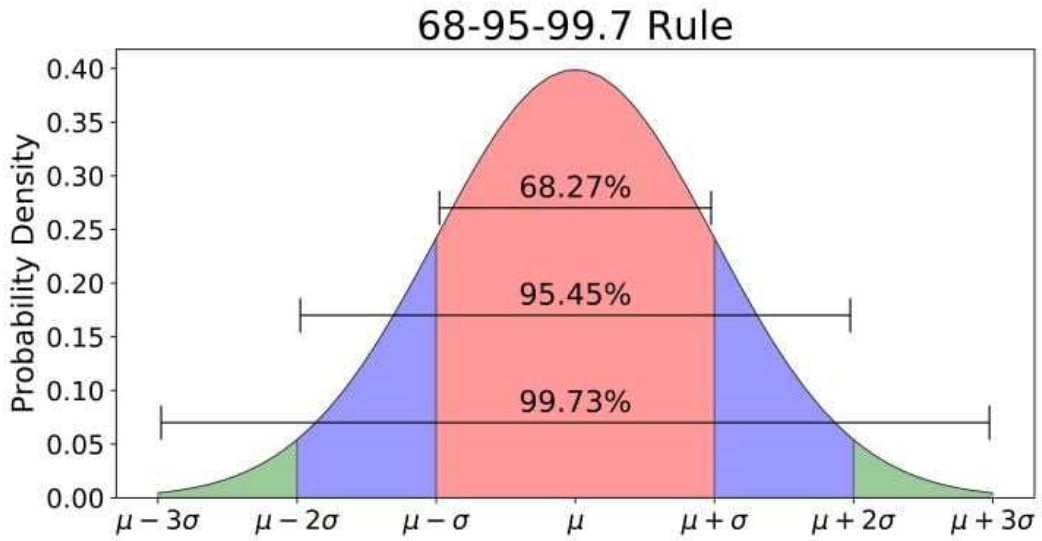
This rule is applicable to normal distributions, which are symmetric and bell-shaped. It's a useful heuristic for quickly estimating the spread of data and identifying outliers or unusual observations.

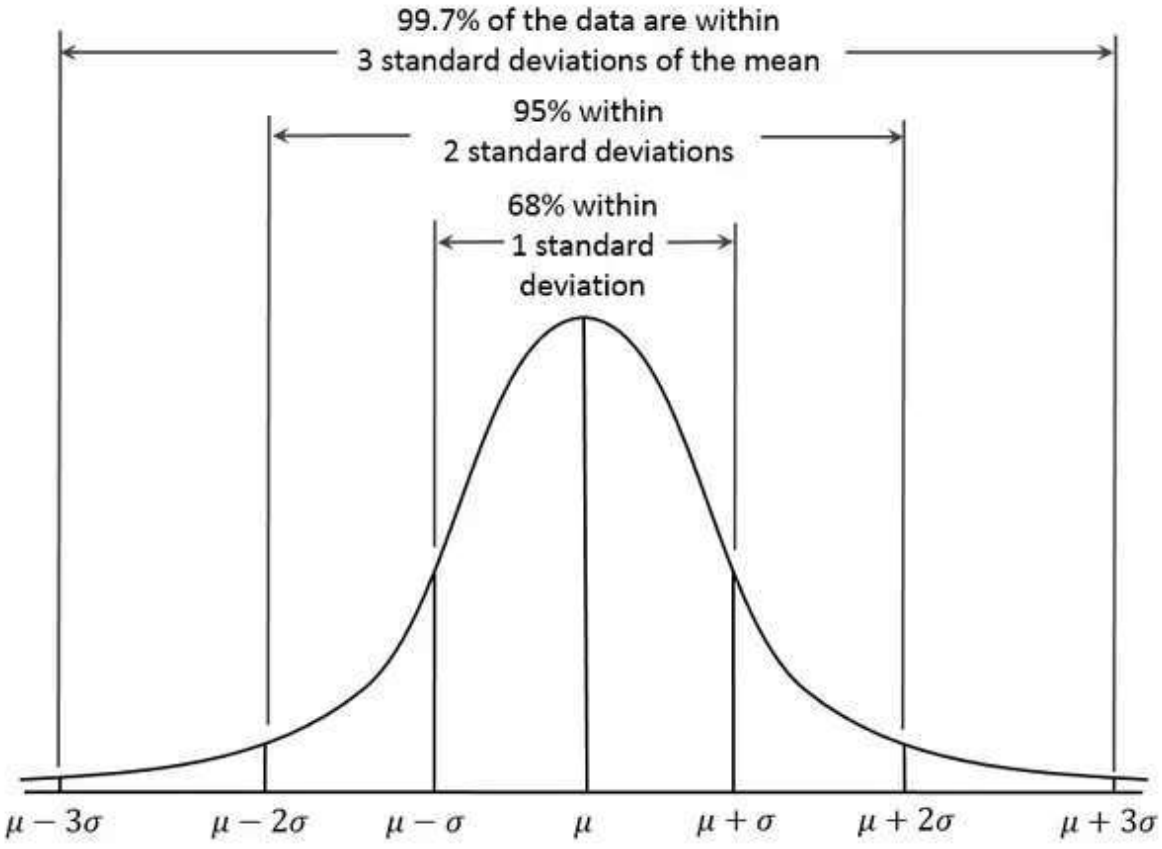
Here's a Python example illustrating the Empirical Rule:

In [30]:

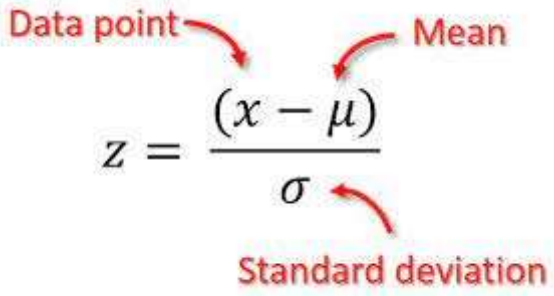
```
import numpy as np

# Generate random data from a normal distribution
np.random.seed(42)  # for reproducibility
mean = 0
std_dev = 1
num_samples = 10000
data = np.random.normal(mean, std_dev, num_samples)

# Calculate the ranges
within_one_std_dev = np.sum(np.logical_and(data >= mean - std_dev, data <=
within_two_std_dev = np.sum(np.logical_and(data >= mean - 2*std_dev, data
within_three_std_dev = np.sum(np.logical_and(data >= mean - 3*std_dev, da

# Calculate the percentages
percentage_within_one_std_dev = within_one_std_dev / num_samples * 100
percentage_within_two_std_dev = within_two_std_dev / num_samples * 100
percentage_within_three_std_dev = within_three_std_dev / num_samples * 10

# Print the results
print("Percentage within one standard deviation: {:.2f}%".format(percentag
print("Percentage within two standard deviations: {:.2f}%".format(percenta
print("Percentage within three standard deviations: {:.2f}%".format(perce


```

```
Percentage within one standard deviation: 68.20%
Percentage within two standard deviations: 95.33%
Percentage within three standard deviations: 99.72%
```





Z-Score: The Z-score is a statistical measure that quantifies how many standard deviations a data point is away from the mean of a dataset. It's commonly used to understand how unusual or typical a data point is within a distribution. The formula to calculate the Z-score is:



In Python, you can calculate the Z-score using various libraries such as NumPy or SciPy. Here's a simple implementation using NumPy:

In [31]:
```
import numpy as np

# Create a dataset
data = np.array([23, 27, 21, 30, 25, 28, 32, 20, 22, 26])

# Calculate mean and standard deviation
mean = np.mean(data)
std_dev = np.std(data)

# Calculate Z-scores for each data point
z_scores = (data - mean) / std_dev

print("Data:", data)
print("Z-scores:", z_scores)

```

```
Data: [23 27 21 30 25 28 32 20 22 26]
Z-scores: [-0.64051262  0.42700841 -1.17427313  1.22764918 -0.1067521   0.693
88867
  1.76140969 -1.44115338 -0.90739287  0.16012815]
```

In this example, we've created a dataset of values and calculated their Z-scores using the formula. The Z-scores indicate how many standard deviations each data point is away from the mean.

Uses of Z-scores in statistics:

Outlier Detection: Z-scores help identify data points that are significantly different from the mean. Points with high Z-scores may indicate outliers.

Standardization: Z-scores are used to standardize data, making it easier to compare and analyze datasets with different scales.

Hypothesis Testing: Z-scores are used to assess the likelihood of an event occurring under a normal distribution, which is often used in hypothesis testing.

Quality Control: Z-scores are used in quality control to monitor and maintain consistent processes.

Risk Assessment: Z-scores are used in finance and risk assessment to measure the risk associated with certain investments or decisions.

Remember that Z-scores are meaningful when your data follows a normal distribution. If your data is not normally distributed, you might consider using other methods like percentiles or other transformations.

Formula to find population mean

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

Formula to find population standard deviation

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

Formula to find the **z-score**

$$z\ score = \frac{(x - \mu)}{\sigma}$$

Probability

Probability is a fundamental concept in statistics that quantifies the likelihood of an event occurring. In Python, you can work with probabilities using various libraries, such as NumPy and SciPy. Let's explore a simple example of calculating probabilities using a fair six-sided die:

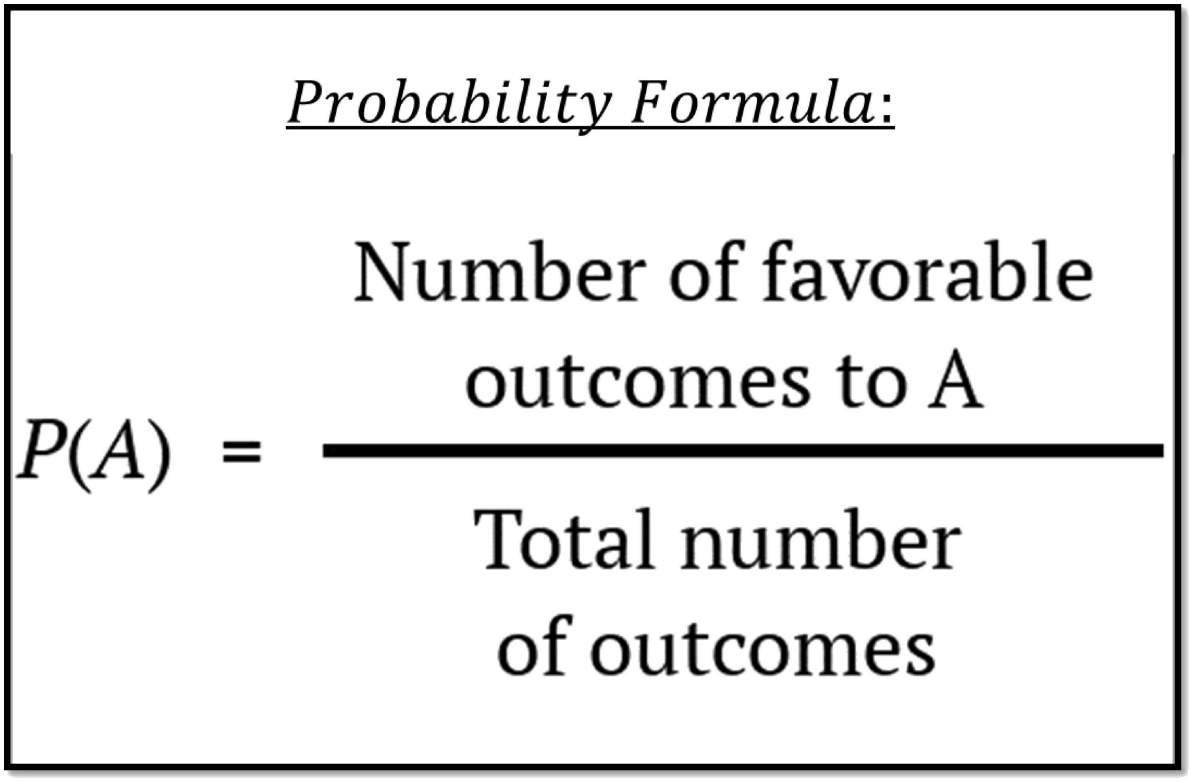
In [32]:
```
import numpy as np

# Simulating a fair six-sided die roll
outcomes = [1, 2, 3, 4, 5, 6]

# Calculate the probability of rolling each number
total_outcomes = len(outcomes)
probabilities = {number: 1 / total_outcomes for number in outcomes}

# Calculate the probability of rolling an even number (2, 4, or 6)
even_numbers = [2, 4, 6]
probability_even = len(even_numbers) / total_outcomes

# Calculate the probability of rolling a number greater than 3 (4, 5, or 6
greater_than_3 = [4, 5, 6]
probability_greater_than_3 = len(greater_than_3) / total_outcomes

# Print the probabilities
print("Individual Probabilities:", probabilities)
print("Probability of even number:", probability_even)
print("Probability of number greater than 3:", probability_greater_than_3)

```

```
Individual Probabilities: {1: 0.16666666666666666, 2: 0.16666666666666666, 3:
0.16666666666666666, 4: 0.16666666666666666, 5: 0.16666666666666666, 6: 0.166
66666666666666}
Probability of even number: 0.5
Probability of number greater than 3: 0.5
```

In this example, we simulate a fair six-sided die roll and calculate various probabilities. The probabilities are calculated by dividing the favorable outcomes by the total number of possible outcomes.

Key concepts covered in the example:

1. Calculating probabilities of individual events.
2. Calculating probabilities of compound events (e.g., rolling an even number or a number greater than 3).



In probability theory, events are subsets of a sample space that represent different possible outcomes or occurrences in an experiment. Here are some common types of events:

1. **Simple Event:** A simple event is a single outcome from a sample space. For example, rolling a specific number on a six-sided die.
2. **Compound Event:** A compound event consists of two or more simple events. It represents a combination of outcomes. For example, rolling an even number (2, 4, or 6) on a six-sided die is a compound event.
3. **Complementary Event:** The complementary event of an event (A) is denoted as (A') or (\bar{A}). It consists of all outcomes not in event (A). For example, if event (A) is rolling an even number on a six-sided die, its complementary event (A') is rolling an odd number.
4. **Mutually Exclusive Events:** Two events are mutually exclusive if they cannot occur at the same time. In other words, if one event occurs, the other cannot. For example, in a single die roll, the events "rolling a 2" and "rolling a 4" are mutually exclusive.
5. **Independent Events:** Two events are independent if the occurrence of one event does not affect the probability of the other event occurring. For example, if you flip a coin twice, the outcomes of the first flip do not influence the outcomes of the second flip.
6. **Dependent Events:** Two events are dependent if the occurrence of one event affects the probability of the other event occurring. For example, drawing cards from a deck without replacement is a situation where events are dependent.
7. **Collectively Exhaustive Events:** A set of events is collectively exhaustive if their union covers the entire sample space. In other words, there are no other possible outcomes. For example, when rolling a six-sided die, the events "rolling an even number" and "rolling an odd number" are collectively exhaustive.
8. **Equally Likely Events:** Events are equally likely if each event has the same probability of occurring. For example, when rolling a fair six-sided die, all six outcomes are equally likely.

9. **Compound Events with Intersection:** The intersection of two events (A) and (B) is denoted as (A \cap B) and represents the set of outcomes that are in both events. For example, if event (A) is rolling an even number and event (B) is rolling a number greater than 3, their intersection (A \cap B) is the outcome 4 or 6.
10. **Compound Events with Union:** The union of two events (A) and (B) is denoted as (A \cup B) and represents the set of outcomes that are in either event (A) or event (B) or both. For example, (A \cup B) includes all even numbers (2, 4, or 6) and all numbers greater than 3 (4, 5, or 6).

Mutual and Non-Mutual Events

Mutual (or mutually exclusive) events and non-mutual (or non-mutually exclusive) events refer to the relationship between events in probability theory.

1. **Mutually Exclusive (Mutual) Events:** Mutually exclusive events are events that cannot occur simultaneously. If one of the events happens, the other event(s) cannot occur. Mathematically, the intersection of mutually exclusive events is an empty set ((A \cap B = \emptyset)).

   For example, consider rolling a six-sided die:

   - Event (A) is rolling an even number ((2, 4, 6)).
   - Event (B) is rolling an odd number ((1, 3, 5)).

   Since a die cannot show both an even and an odd number simultaneously, events (A) and (B) are mutually exclusive.
2. **Non-Mutually Exclusive (Non-Mutual) Events:** Non-mutually exclusive events are events that can occur together. They have at least one outcome in common. Mathematically, the intersection of non-mutually exclusive events is not an empty set ((A \cap B \neq \emptyset)).

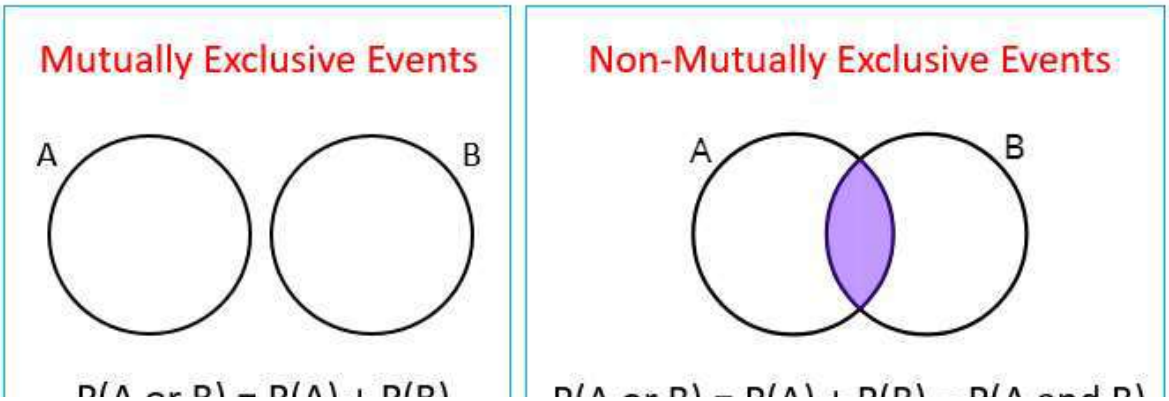
   For example, consider drawing cards from a standard deck:

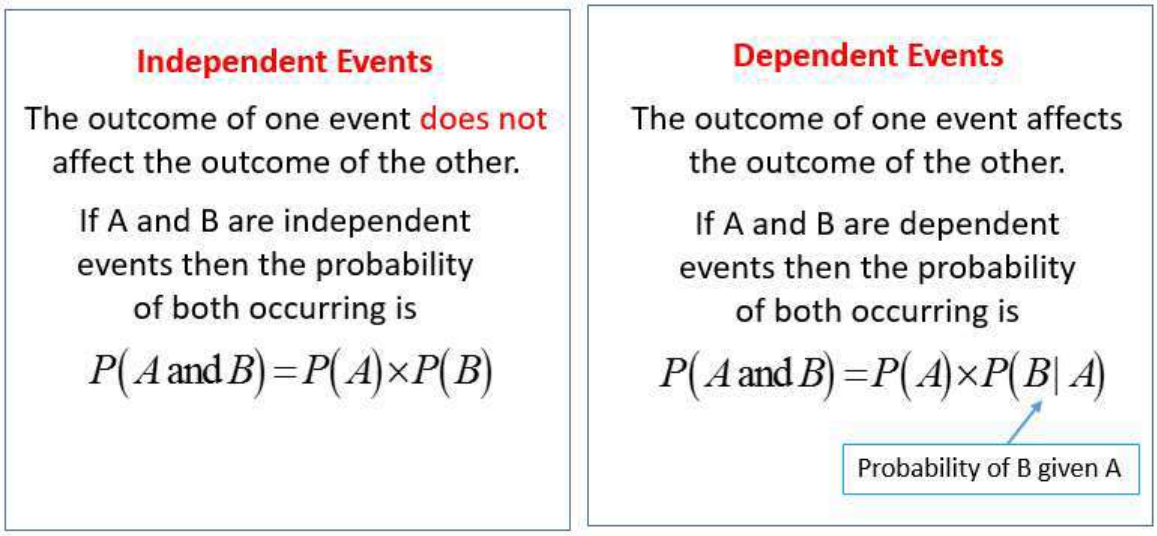
   - Event (C) is drawing a red card.
   - Event (D) is drawing a heart card.

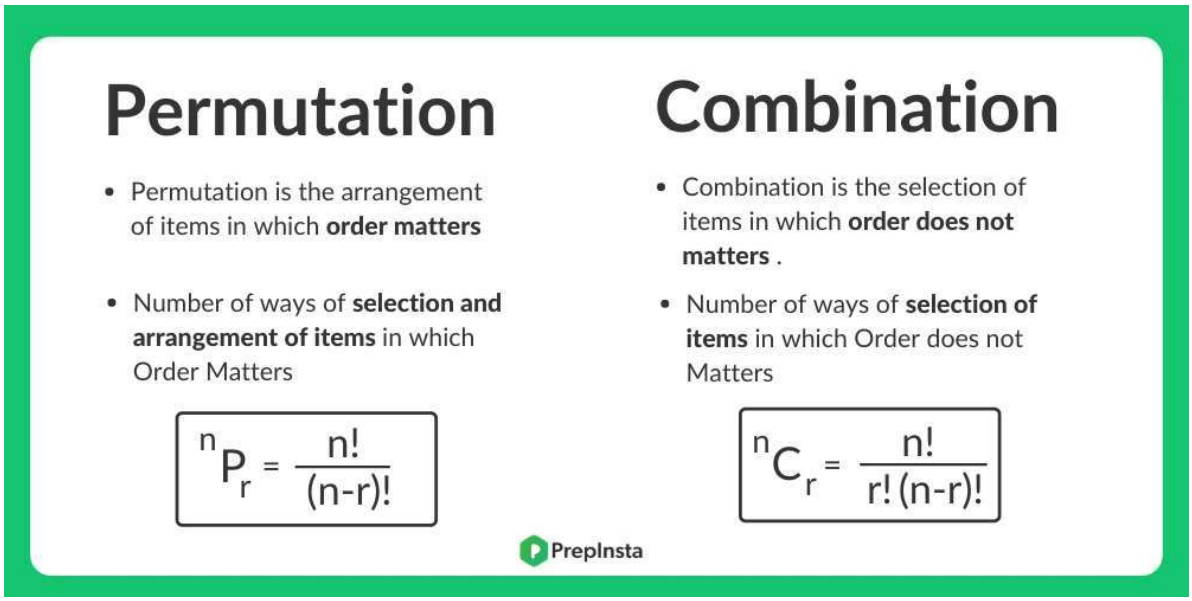
   Both events (C) and (D) can occur together if you draw a red heart card. Therefore, events (C) and (D) are non-mutually exclusive.

The distinction between mutually exclusive and non-mutually exclusive events is important in probability calculations, especially when calculating probabilities of combined or compound events. When events are mutually exclusive, you can use the addition rule to calculate the probability of either event occurring ((P(A \cup B) = P(A) + P(B))). When events are not mutually exclusive, you need to account for the overlap in calculating the combined probability ((P(A \cup B) = P(A) + P(B) - P(A \cap B))).

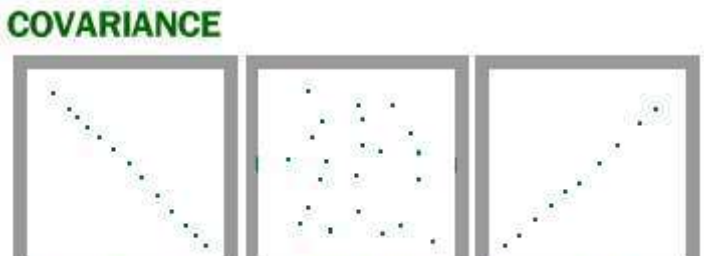
Understanding whether events are mutually exclusive or not helps in correctly analyzing and calculating probabilities in various scenarios.|







Covariance: Covariance is a statistical measure that quantifies the degree to which two variables change together. It indicates whether an increase in one variable corresponds to an increase or decrease in another variable. It's calculated using the formula:



Population Covariance Formula

$$Cov(x,y) = \frac{\Sigma(x_i-\bar{x})(y_i-\bar{y})}{N}$$

Sample Covariance

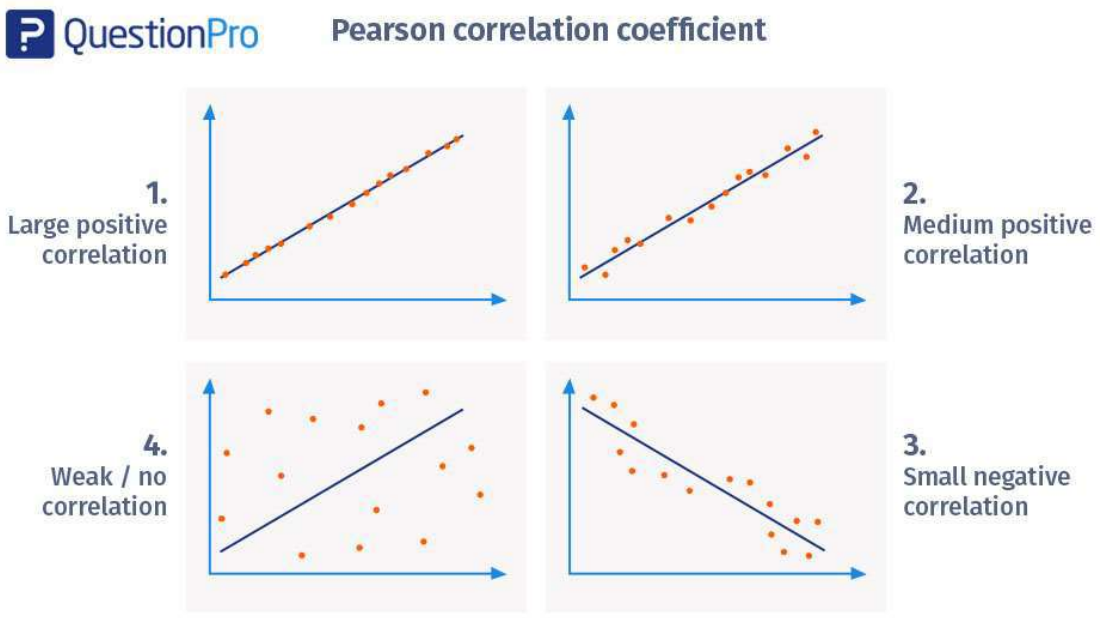
$$Cov(x,y) = \frac{\Sigma(x_i-\bar{x})(y_i-y)}{N-1}$$

A positive covariance indicates that the variables tend to increase together, while a negative covariance indicates that one variable tends to decrease as the other increases.

Correlation: Correlation measures the strength and direction of the linear relationship between two variables. It's a standardized measure that ranges from -1 to 1. A correlation value of 1 indicates a perfect positive linear relationship, -1 indicates a perfect negative linear relationship, and 0 indicates no linear relationship.

Correlation Formula

Pearson's Correlation: Pearson's correlation coefficient, often denoted as (r), measures the linear relationship between two continuous variables. It's calculated using the formula:



Spearman's Rank Correlation: Spearman's rank correlation coefficient, often denoted as (r_s), assesses the strength and direction of the monotonic relationship (not necessarily linear) between two variables. It's based on the ranks of the data points rather than the actual values.

Here's Python code to calculate covariance, Pearson's correlation, and Spearman's rank correlation using NumPy and SciPy:

In [34]:
```python
import numpy as np
from scipy.stats import pearsonr, spearmanr

# Sample data
x = np.array([12, 18, 20, 24, 30])
y = np.array([35, 30, 40, 55, 50])

# Calculate covariance
covariance = np.cov(x, y)[0][1]

# Calculate Pearson's correlation
pearson_corr, _ = pearsonr(x, y)

# Calculate Spearman's rank correlation
spearman_corr, _ = spearmanr(x, y)

print("Covariance:", covariance)
print("Pearson's Correlation:", pearson_corr)
print("Spearman's Rank Correlation:", spearman_corr)

```

```
Covariance: 53.0
Pearson's Correlation: 0.7603305138058882
Spearman's Rank Correlation: 0.7999999999999999
```

Hypothesis testing

Hypothesis testing is a fundamental concept in statistics that allows us to make informed decisions about population parameters based on sample data. It involves evaluating two competing statements, called the null hypothesis (H0) and the alternative hypothesis (Ha or H1), using statistical methods.

Here's a step-by-step breakdown of the hypothesis testing process:

1. **State the Hypotheses:**
   - Null Hypothesis (H0): This is the default assumption that there is no significant effect or difference. It often states that any observed difference is due to random chance.
   - Alternative Hypothesis (Ha or H1): This is the claim you want to test. It states that there is a significant effect or difference in the data.
2. **Choose a Significance Level (Alpha):**
   - The significance level ((\alpha)) is the probability of rejecting the null hypothesis when it's actually true. It determines the threshold for considering evidence strong enough to reject the null hypothesis.
3. **Collect and Analyze Data:**
   - Collect a representative sample from the population.
   - Calculate sample statistics such as means, proportions, etc.
4. **Calculate the Test Statistic:**
   - The test statistic is a value that summarizes the data and is used to determine the likelihood of observing the data if the null hypothesis is true.

- The choice of test statistic depends on the type of test you're conducting (e.g., t-test, chi-squared test, etc.).

5. **Determine the Critical Region:**

   - The critical region is the range of values of the test statistic that would lead to rejecting the null hypothesis.
   - The critical region is determined by the significance level and the distribution of the test statistic (e.g., t-distribution for t-tests, chi-squared distribution for chi-squared tests).

6. **Calculate the P-value:**

   - The p-value is the probability of observing data as extreme as, or more extreme than, the data actually observed, assuming that the null hypothesis is true.
   - A small p-value indicates strong evidence against the null hypothesis.

7. **Make a Decision:**

   - If the p-value is less than or equal to the significance level ((p \leq \alpha)), you reject the null hypothesis in favor of the alternative hypothesis.
   - If the p-value is greater than the significance level ((p > \alpha)), you fail to reject the null hypothesis.

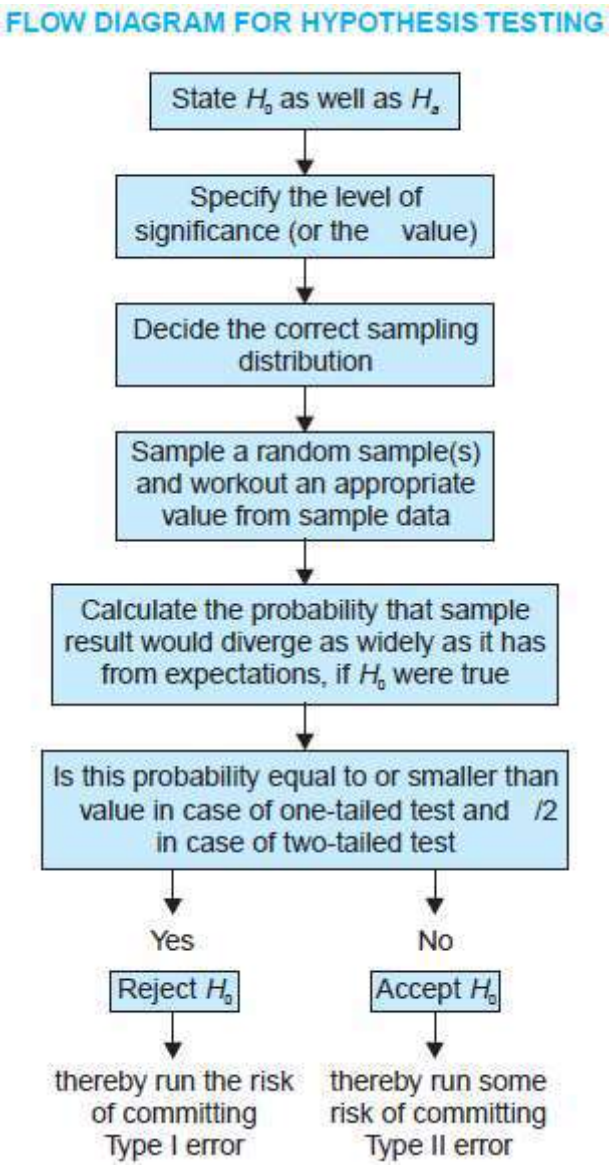
8. **Draw a Conclusion:**

   - If you reject the null hypothesis, you conclude that there is evidence to support the alternative hypothesis.
   - If you fail to reject the null hypothesis, you do not have enough evidence to support the alternative hypothesis.

9. **Report the Results:**

   - Clearly state whether you rejected or failed to reject the null hypothesis.
   - Provide the p-value and the significance level used.

Hypothesis testing is used in a wide range of fields to make data-driven decisions, validate assumptions, and draw conclusions about populations based on sample data. The choice of hypothesis test depends on the nature of the data, the research question, and the hypotheses being tested. Common tests include t-tests, chi-squared tests, ANOVA (analysis of variance), and regression analysis.



Common tests in Hypothesis testing:

T-Tests: A t-test is a statistical test used to determine whether there is a significant difference between the means of two groups. There are different types of t-tests based on the characteristics of the data and the research question. Here, we'll cover the Independent Samples t-test, which is used when comparing means of two independent groups.

Example: Independent Samples T-Test

In [35]:
```
import numpy as np
from scipy import stats

# Sample data for two groups
group1 = np.array([56, 60, 62, 58, 54])
group2 = np.array([65, 68, 70, 67, 72])

# Perform independent samples t-test
t_statistic, p_value = stats.ttest_ind(group1, group2)

# Print the results
print("T-Statistic:", t_statistic)
print("P-Value:", p_value)

# Interpret the results
if p_value < 0.05:
    print("Reject the null hypothesis: There is a significant difference b
else:
    print("Fail to reject the null hypothesis: There is no significant di

```

```
T-Statistic: -5.591074637932895
P-Value: 0.0005155874864632357
Reject the null hypothesis: There is a significant difference between the gro
up means.
```

Chi-Squared Test: The chi-squared test is used to determine whether there is a significant association between two categorical variables. It compares the observed frequency distribution with the expected frequency distribution under the assumption of independence.

In [36]:

```
import numpy as np
from scipy.stats import chi2_contingency

# Observed frequency data in a contingency table
observed = np.array([[30, 20], [10, 40]])

# Perform chi-squared test
chi2_stat, p_value, dof, expected = chi2_contingency(observed)

# Print the results
print("Chi-Squared Statistic:", chi2_stat)
print("P-Value:", p_value)
print("Degrees of Freedom:", dof)
print("Expected Frequencies:\n", expected)

# Interpret the results
if p_value < 0.05:
    print("Reject the null hypothesis: There is a significant association
else:
    print("Fail to reject the null hypothesis: There is no significant ass

```

```
Chi-Squared Statistic: 15.041666666666668
P-Value: 0.00010516355403363114
Degrees of Freedom: 1
Expected Frequencies:
 [[20. 30.]
 [20. 30.]]
Reject the null hypothesis: There is a significant association between the va
riables.
```

ANOVA (Analysis of Variance): ANOVA is used to compare means among three or more groups. It tests whether there are any statistically significant differences between the group means. One-way ANOVA compares means across a single factor, while two-way ANOVA considers two factors.

Example: One-Way ANOVA

In [37]:
```
import numpy as np
from scipy.stats import f_oneway

# Sample data for three groups
group1 = np.array([54, 58, 60, 62, 57])
group2 = np.array([65, 68, 70, 67, 72])
group3 = np.array([75, 78, 80, 77, 82])

# Perform one-way ANOVA
f_statistic, p_value = f_oneway(group1, group2, group3)

# Print the results
print("F-Statistic:", f_statistic)
print("P-Value:", p_value)

# Interpret the results
if p_value < 0.05:
    print("Reject the null hypothesis: There are significant differences b
else:
    print("Fail to reject the null hypothesis: There are no significant di

```

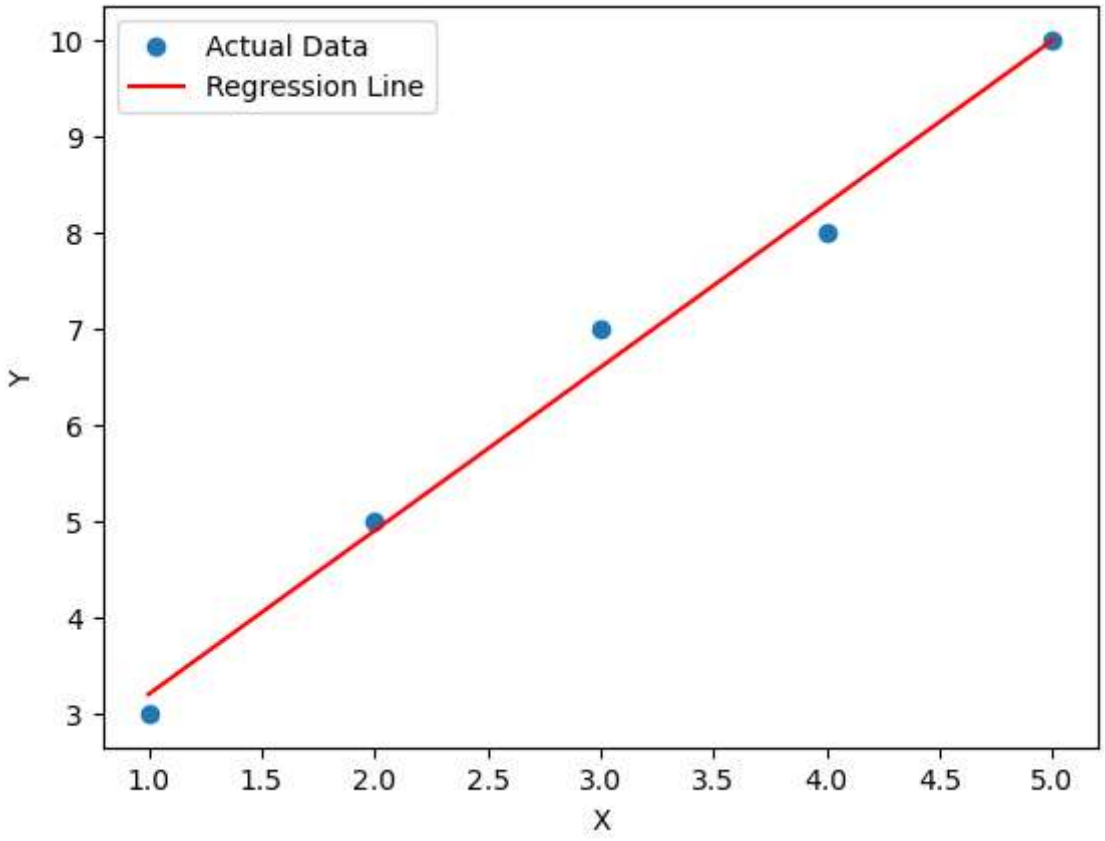
```
F-Statistic: 64.29411764705887
P-Value: 3.867172981913934e-07
Reject the null hypothesis: There are significant differences between the gro
up means.
```

Regression Analysis: Regression analysis is used to model the relationship between one or more independent variables and a dependent variable. Linear regression models a linear relationship, while nonlinear regression models more complex relationships.

Example: Linear Regression

In [38]:

```
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

# Sample data
x = np.array([1, 2, 3, 4, 5])
y = np.array([3, 5, 7, 8, 10])

# Reshape data for sklearn
x = x.reshape(-1, 1)

# Create a linear regression model
model = LinearRegression()
model.fit(x, y)

# Predict y values using the model
y_pred = model.predict(x)

# Plot the data and regression line
plt.scatter(x, y, label="Actual Data")
plt.plot(x, y_pred, color='red', label="Regression Line")
plt.xlabel("X")
plt.ylabel("Y")
plt.legend()
plt.show()

# Print the regression equation
print("Regression Equation: y =", model.coef_[0], "x +", model.intercept_
```



```
Regression Equation: y = 1.7000000000000004 x + 1.4999999999999982
```

These are just basic examples to introduce you to the concepts of these statistical tests and analyses. Depending on the complexity of your data and research questions, you might need to explore more advanced techniques and analyses.

Central limit theorem

The Central Limit Theorem (CLT) is a fundamental concept in statistics that states that the sampling distribution of the sample means will be approximately normally distributed regardless of the original distribution of the population, as long as the sample size is sufficiently large.

Here's a more detailed explanation of the Central Limit Theorem along with an example:

Central Limit Theorem Explanation:

1. **Original Population:** Consider a population with any distribution, which could be normal, skewed, or any other shape.
2. **Sample Means:** If we take random samples of a certain size (n) from this population and calculate the mean of each sample, we create a new distribution of sample means.
3. **Sampling Distribution:** The Central Limit Theorem states that as the sample size increases, the distribution of these sample means will approach a normal distribution.
4. **Properties of the New Distribution:**
    - The mean of the sample means will be very close to the mean of the original population.
    - The standard deviation of the sample means (standard error) will be smaller than the standard deviation of the original population divided by the square root of the sample size.

Example of Central Limit Theorem:

Let's use a practical example to demonstrate the Central Limit Theorem. Consider rolling a fair six-sided die.

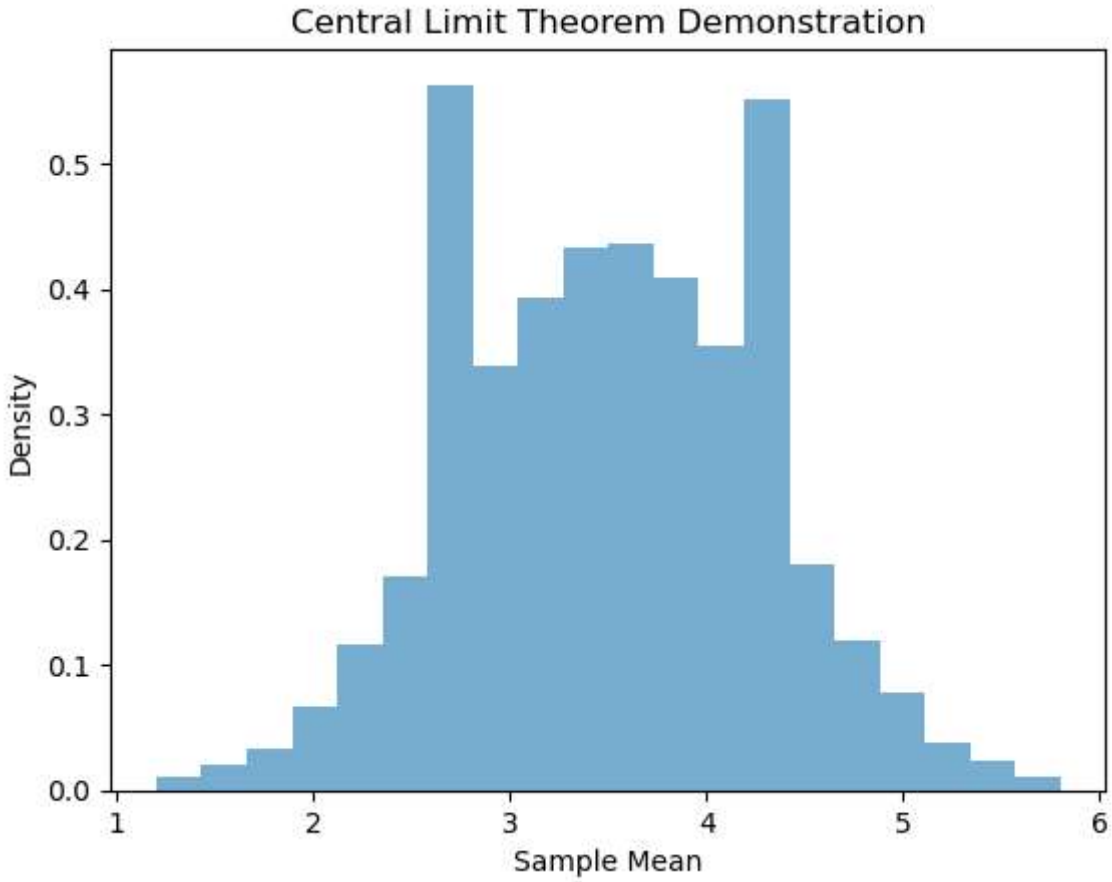
1. **Original Population:** The population consists of all possible outcomes when rolling the die: ({1, 2, 3, 4, 5, 6}).
2. **Sample Means:** We'll take samples of size (n = 5) and calculate the mean for each sample.
3. **Sampling Distribution:** We'll repeat this process for a large number of samples and create a histogram of the sample means.

In this example, even though the original population distribution is discrete, the distribution of the sample means starts to resemble a normal distribution as the sample size increases. This is a manifestation of the Central Limit Theorem in action.

The Central Limit Theorem is particularly useful in inferential statistics, allowing us to make assumptions about population parameters using sample statistics, even when we don't know the exact population distribution.

In [39]:

```python
import numpy as np
import matplotlib.pyplot as plt

# Parameters
population = np.array([1, 2, 3, 4, 5, 6])
sample_size = 5
num_samples = 10000

# Create an array to store sample means
sample_means = []

# Generate sample means
for _ in range(num_samples):
    sample = np.random.choice(population, size=sample_size)
    sample_mean = np.mean(sample)
    sample_means.append(sample_mean)

# Plot the histogram of sample means
plt.hist(sample_means, bins=20, density=True, alpha=0.6)
plt.xlabel("Sample Mean")
plt.ylabel("Density")
plt.title("Central Limit Theorem Demonstration")
plt.show()

```



Distributions in statistics describe the way data is spread out and the likelihood of observing different values. There are various types of distributions, each with its own characteristics and applications. Here are some of the key types of distributions:

1. **Normal Distribution (Gaussian Distribution):**
    - The normal distribution is symmetrical and bell-shaped, with the mean, median, and mode all coinciding at the center.
    - It's characterized by the parameters mean (μ) and standard deviation (σ).
    - Many natural phenomena, such as heights and weights, tend to follow a normal distribution.
    - The Central Limit Theorem states that the distribution of sample means from any population approaches a normal distribution as the sample size increases.
2. **Binomial Distribution:**
    - The binomial distribution models the number of successes in a fixed number of independent Bernoulli trials, where each trial has a constant probability of success (p).
    - It's characterized by two parameters: the number of trials (n) and the probability of success (p).
    - It's used to model events with two possible outcomes, like coin flips or pass/fail scenarios.
3. **Poisson Distribution:**
    - The Poisson distribution models the number of events that occur in a fixed interval of time or space, given a constant average rate of occurrence (λ).
    - It's characterized by the parameter λ.
    - It's used for counting the occurrences of rare events, such as the number of phone calls received at a call center within a specific time period.
4. **Exponential Distribution:**
    - The exponential distribution models the time between events in a Poisson process (events occurring at a constant average rate).
    - It's characterized by the parameter λ (average rate of occurrence).
    - It's used in reliability analysis and queuing theory.
5. **Uniform Distribution:**
    - The uniform distribution has a constant probability for all values within a specified range.
    - It's characterized by two parameters: the minimum and maximum values of the range.
    - It's used when each value within a range is equally likely, like rolling a fair six-sided die.
6. **Gamma Distribution:**
    - The gamma distribution generalizes the exponential distribution and models the time until (k) events occur in a Poisson process.
    - It's characterized by two parameters: shape (k) and scale (θ).
    - It's used in various fields including queuing theory, reliability analysis, and finance.
7. **Chi-Squared Distribution:**
    - The chi-squared distribution arises from the sum of the squares of (k) independent standard normal random variables.
    - It's often used in hypothesis testing, especially in tests involving categorical data.
8. **Student's t-Distribution:**

- The t-distribution is used when the sample size is small and the population standard deviation is unknown.
- It's characterized by the degrees of freedom ((df)) and resembles the normal distribution but has heavier tails.

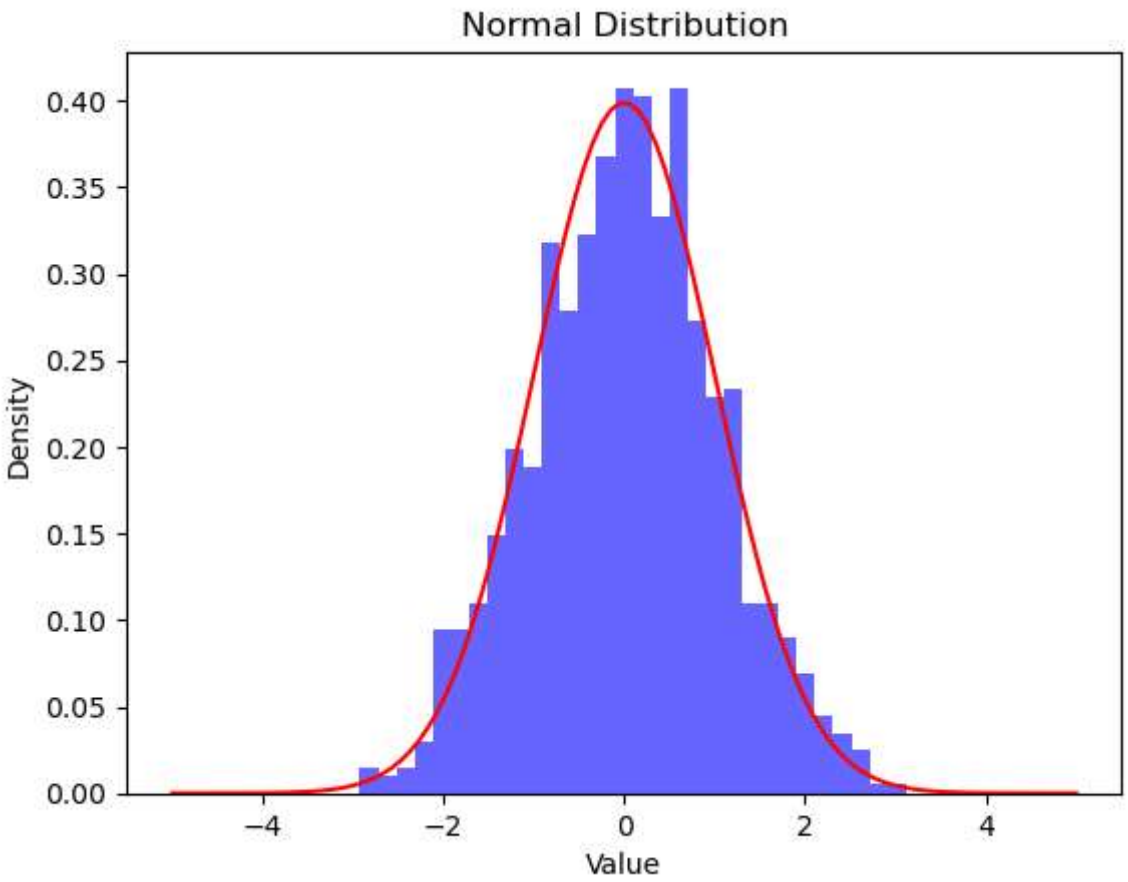
9. **F-Distribution:**

   - The F-distribution arises in the context of comparing variances from two or more populations.
   - It's characterized by two degrees of freedom parameters and is commonly used in ANOVA and regression analysis.

These are just a few of the many distributions in statistics. The choice of distribution depends on the nature of the data and the specific problem you are trying to solve.

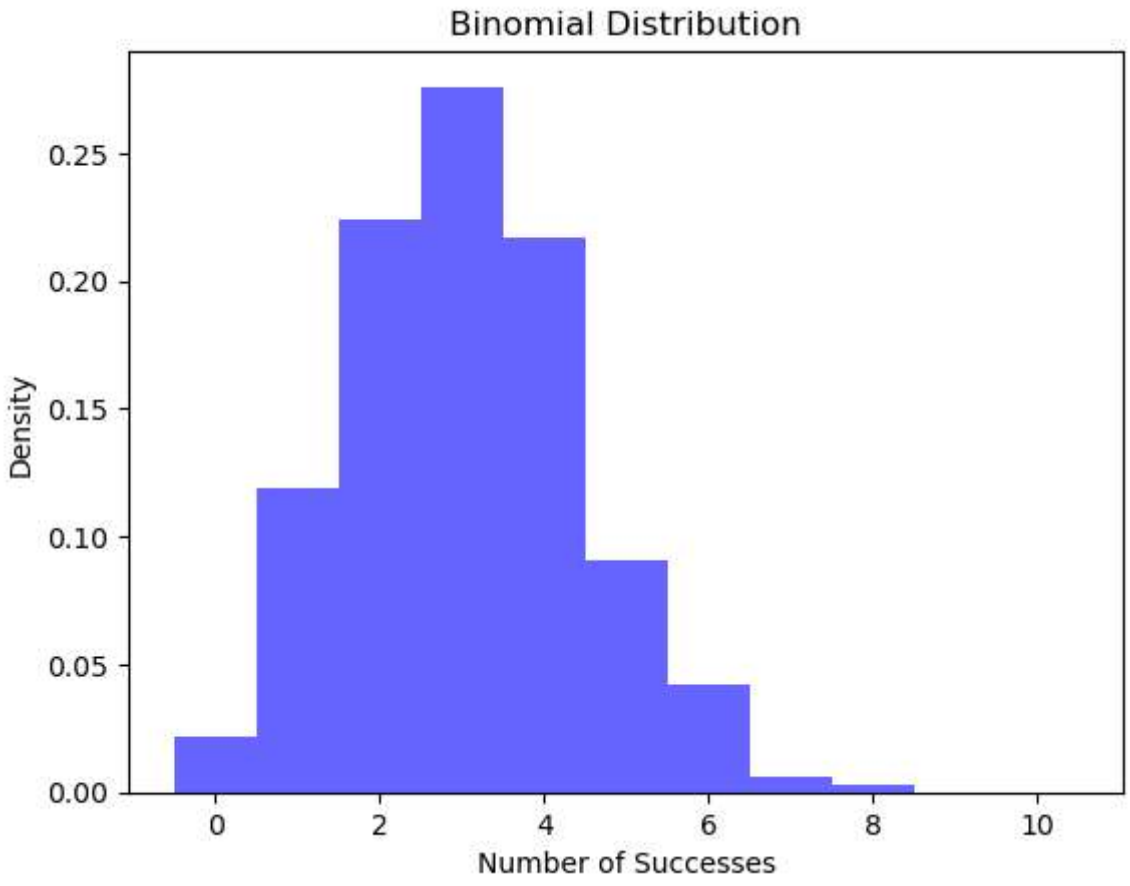
In [40]:

```
# Normal Distribution

import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

# Parameters for the normal distribution
mean = 0
std_dev = 1

# Generate random data from a normal distribution
data = np.random.normal(mean, std_dev, 1000)

# Create a histogram to visualize the distribution
plt.hist(data, bins=30, density=True, alpha=0.6, color='blue')

# Plot the normal distribution curve
x = np.linspace(-5, 5, 100)
y = norm.pdf(x, mean, std_dev)
plt.plot(x, y, color='red')

plt.xlabel('Value')
plt.ylabel('Density')
plt.title('Normal Distribution')
plt.show()

```



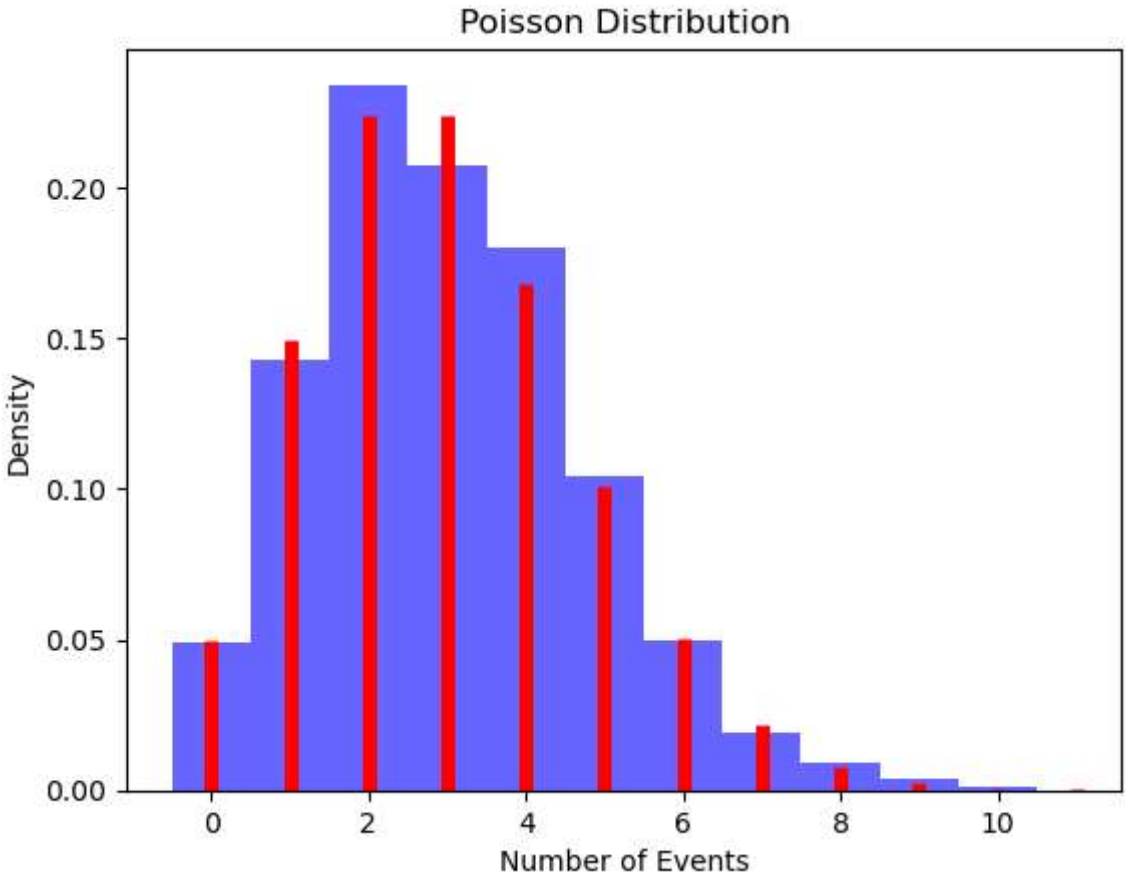
In [41]:

```
# Binomial Distribution

import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import binom

# Parameters for the binomial distribution
n = 10  # Number of trials
p = 0.3  # Probability of success

# Generate random data from a binomial distribution
data = np.random.binomial(n, p, 1000)

# Create a histogram to visualize the distribution
plt.hist(data, bins=np.arange(0, n + 2) - 0.5, density=True, alpha=0.6, c

plt.xlabel('Number of Successes')
plt.ylabel('Density')
plt.title('Binomial Distribution')
plt.show()

```



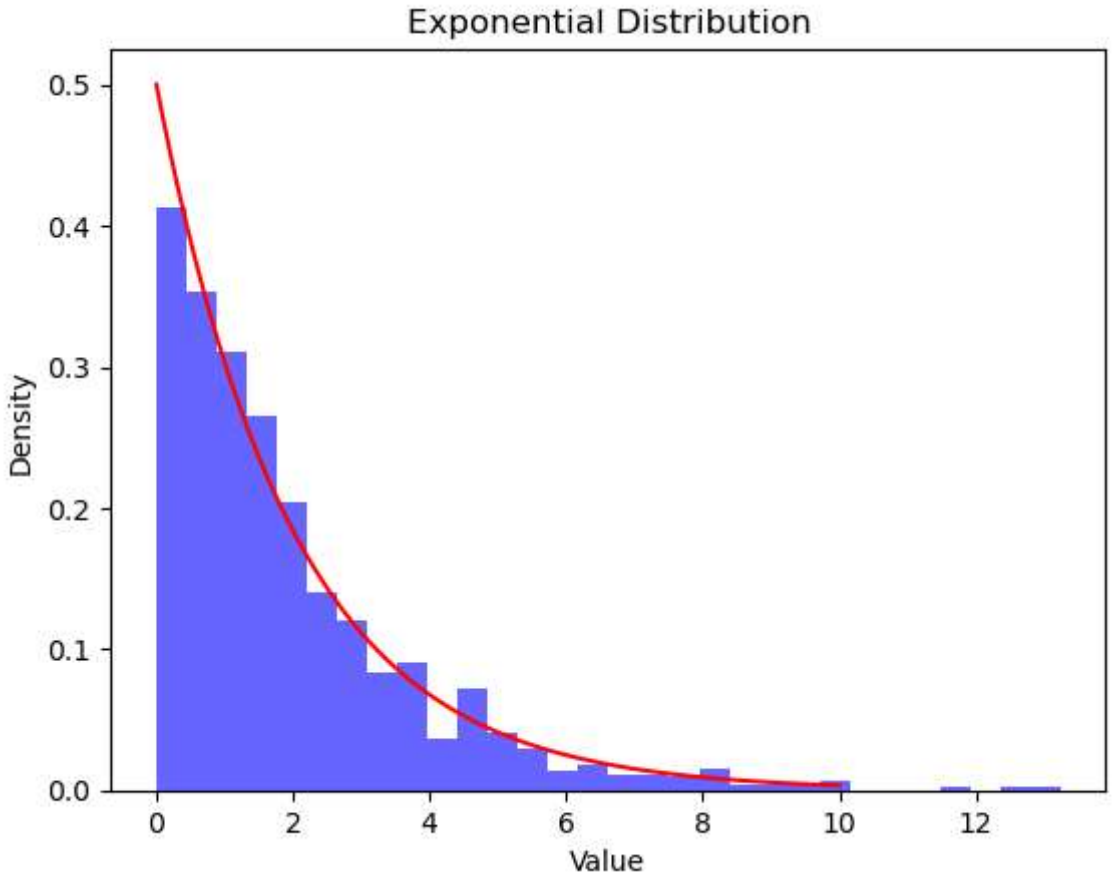
In [42]:

```
# Possion Distribution

import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import poisson

# Parameter for the Poisson distribution
lambda_ = 3

# Generate random data from a Poisson distribution
data = np.random.poisson(lambda_, 1000)

# Create a histogram to visualize the distribution
plt.hist(data, bins=np.arange(0, 12) - 0.5, density=True, alpha=0.6, colo

x = np.arange(0, 12)
y = poisson.pmf(x, lambda_)
plt.vlines(x, 0, y, colors='red', lw=5)

plt.xlabel('Number of Events')
plt.ylabel('Density')
plt.title('Poisson Distribution')
plt.show()

```



In [43]:

```
#  Exponential Distribution

import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import expon

# Parameter for the exponential distribution
lambda_ = 0.5

# Generate random data from an exponential distribution
data = np.random.exponential(scale=1/lambda_, size=1000)

# Create a histogram to visualize the distribution
plt.hist(data, bins=30, density=True, alpha=0.6, color='blue')

x = np.linspace(0, 10, 100)
y = expon.pdf(x, scale=1/lambda_)
plt.plot(x, y, color='red')

plt.xlabel('Value')
plt.ylabel('Density')
plt.title('Exponential Distribution')
plt.show()

```



In [44]:
```
1
```

In [ ]:
```
1
```