Statistics for data science

What is Statistics?

- Statistics is the study of the collection, analysis, interpretation, presentation, and organization of data.
- In other words, it is a mathematical discipline to collect, summarize data.

Statistics Terminologies

Some of the most common terms you might come across in statistics are:

Population:

- It is a collection of a set of individual objects or events whose properties are to be analysed.

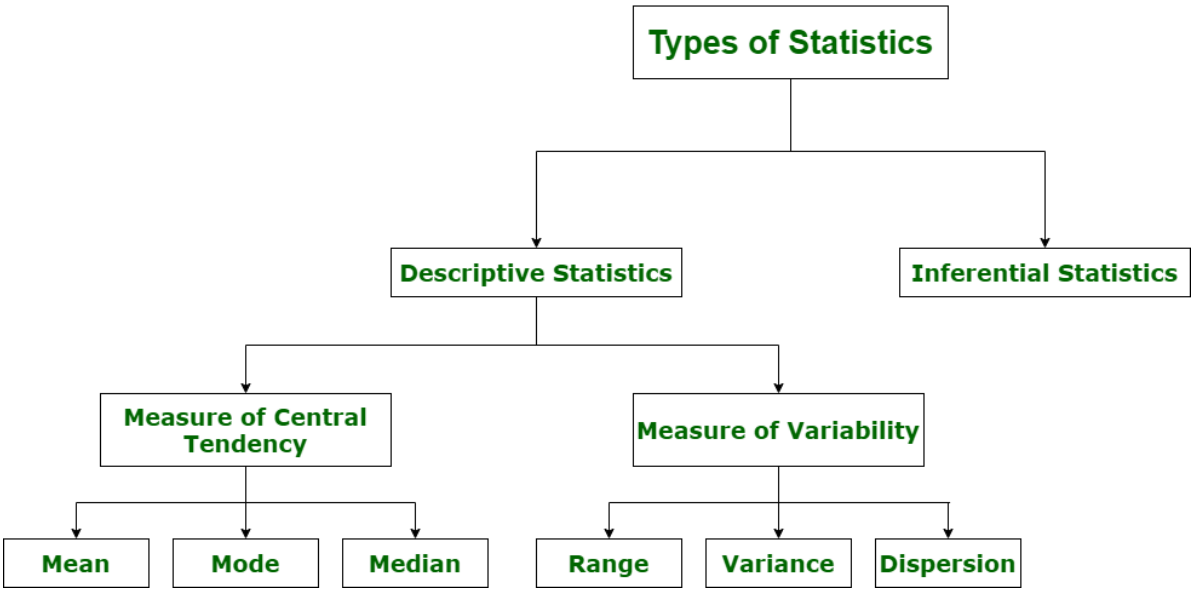
Sample:

- It is the subset of a population.

Types of Statistics

There are 2 types of statistics:

- Descriptive Statistics
- Inferential Statistics



Descriptive Statistics

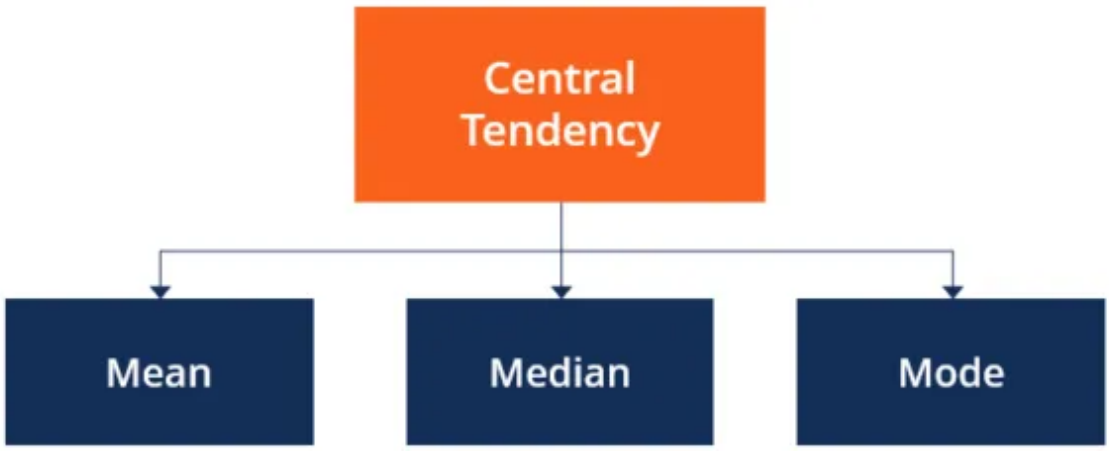
Descriptive statistics are a way of summarizing and describing the main features of a dataset.

They help us understand and communicate key characteristics of the data without going into complex details.

There are two categories in this as follows.

- Measure of Central Tendency
- Measure of Variability

Measure of Central Techency



In [ ]:

Mean

- Mean (Average): The sum of all values divided by the number of values.

$$\text{Mean } (x) = \frac{\sum x}{n}$$

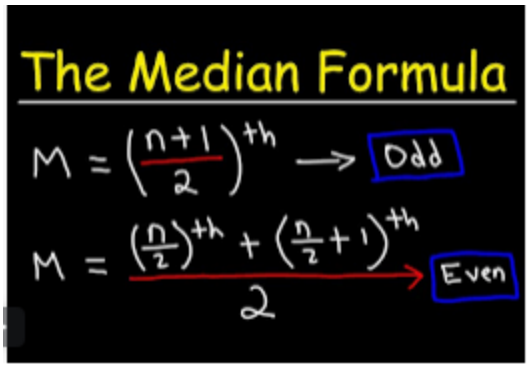
Mean Formula

In [25]:
```
data = [5, 7, 8, 10, 12, 15, 17, 19, 21, 23]

mean = sum(data) / len(data)
print(f"Mean: {mean}")
```

```
Mean: 13.7
```

Median

The middle value when the data is arranged in ascending order.



In [29]:
```
data = [5, 7, 8, 10, 12, 15, 17, 18, 20, 21, 25]

sorted_data = sorted(data)
n = len(sorted_data)

if n % 2 == 0:
    median = (sorted_data[n // 2 - 1] + sorted_data[n // 2]) / 2
```

```
else:
    median = sorted_data[n // 2]

print(f"Median: {median}")
```

```
Median: 15
```

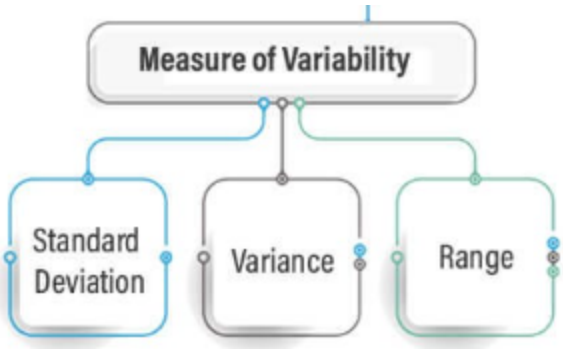
Mode

The value that appears most frequently.

In [31]:
```
from statistics import mode

# Sample list of numbers
numbers = [1, 2, 2, 3, 3, 3, 4, 4, 5, 3, 2, 2, 2, 5, 6, 8, 4, 4, 4, 4,]

# Calculate the mode
result = mode(numbers)

# Print the result
print("Mode:", result)
```

```
Mode: 4
```

Measures of dispersion



Range

The difference between the maximum and minimum values.

$$R = H - L$$

- R = range
- H = highest value
- L = lowest value

In [33]:
```
data = [9, 5, 7, 8, 10, 12, 15, 45, 60, 1]

data_range = max(data) - min(data)
print(f"Range: {data_range}")
```

```
Range: 59
```

Variance

Variance measures how much each number in the dataset differs from the mean, squared.

Formula	Explanation
$s^2 = \frac{\sum(X - \bar{x})^2}{n - 1}$	• s^2 = sample variance • $\sum$ = sum of... • X = each value • $\bar{x}$ = sample mean • n = number of values in the sample

In [35]:
```
import statistics

data = [5, 7, 8, 10, 12, 15, 17, 19, 21, 23, 25, 27, 34]

# Calculate the variance
variance = statistics.variance(data)

print(f"Variance: {variance}")
```

```
Variance: 75.97435897435898
```

Standard Deviation

The square root of the variance.

Formula	Explanation
$s = \sqrt{\frac{\sum (X - \bar{x})^2}{n - 1}}$	• s = sample standard deviation • $\sum$ = sum of... • X = each value • $\bar{x}$ = sample mean • n = number of values in the sample

In [36]:
```
import statistics

data = [5, 7, 8, 10, 12, 15, 17, 19, 21, 23, 25, 27, 34]
```

```
std_dev = statistics.stdev(data)
print(f"Standard Deviation: {std_dev}")
```

```
Standard Deviation: 8.716327149342145
```

Percentiles

- Percentiles divide a dataset into 100 equal parts, where each part represents a percentage of the data below it.
- For instance, the 75th percentile represents the value below which 75% of the data falls.

P = (n/N) × 100

Where,

- n = ordinal rank of the given value or value below the number
- N = number of values in the data set
- P = percentile

Example:

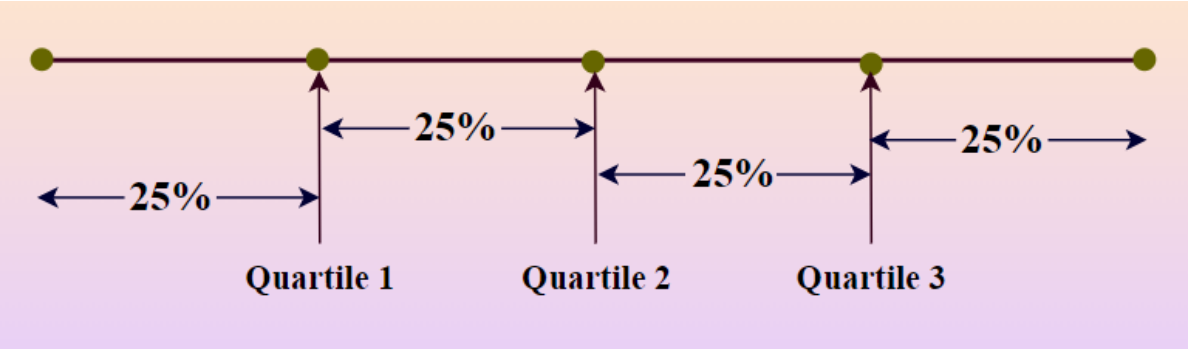
- If you scored at the 75th percentile in a test, it means you scored better than 75% of the people who took the test.

In [41]:
```
import numpy as np

data = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])

# Calculate the 75th percentile
percentile_75 = np.percentile(data, 75)

print(f"75th Percentile: {percentile_75}")
```

```
75th Percentile: 11.5
```

Quartiles

- Quartiles are three values that split your dataset into quarters.
- The first quartile (Q1) represents the 25th percentile, the second quartile (Q2) represents the median (50th percentile), and the third quartile (Q3) represents the 75th percentile.



Example:

- If you're looking at Q2 (the second quartile), it represents the value below which 50% of the data falls.

In [43]:
```
import numpy as np

#data = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
data = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])

# Calculate quartiles
quartiles = np.percentile(data, [25, 50, 75])

Q1, Q2, Q3 = quartiles

print(f"Q1 (25th Percentile): {Q1}")
print(f"Q2 (50th Percentile - Median): {Q2}")
print(f"Q3 (75th Percentile): {Q3}")
```

```
Q1 (25th Percentile): 4.5
Q2 (50th Percentile - Median): 8.0
Q3 (75th Percentile): 11.5
```

Interquartile Range

- The interquartile range defines the difference between the third and the first quartile. ##### or
- The difference between the upper and lower quartile is known as the interquartile range.

Interquartile Range Formula

$$\text{Interquartile range} = \text{Upper Quartile} - \text{Lower Quartile} = Q_3 - Q_1$$

In [44]:
```
import numpy as np

# Sample dataset
data = [12, 15, 17, 20, 22, 25, 28, 30, 32, 35, 40]

# Calculate the first and third quartiles
q1 = np.percentile(data, 25)
```

```
q3 = np.percentile(data, 75)

# Calculate the interquartile range (IQR)
iqr = q3 - q1

print("First Quartile (Q1):", q1)
print("Third Quartile (Q3):", q3)
print("Interquartile Range (IQR):", iqr)
```

```
First Quartile (Q1): 18.5
Third Quartile (Q3): 31.0
Interquartile Range (IQR): 12.5
```

In [ ]:

Skewness:

- Skewness quantifies the degree and direction of skew (departure from horizontal symmetry) in a dataset.
- A negative skewness indicates a longer left tail, while a positive skewness indicates a longer right tail.

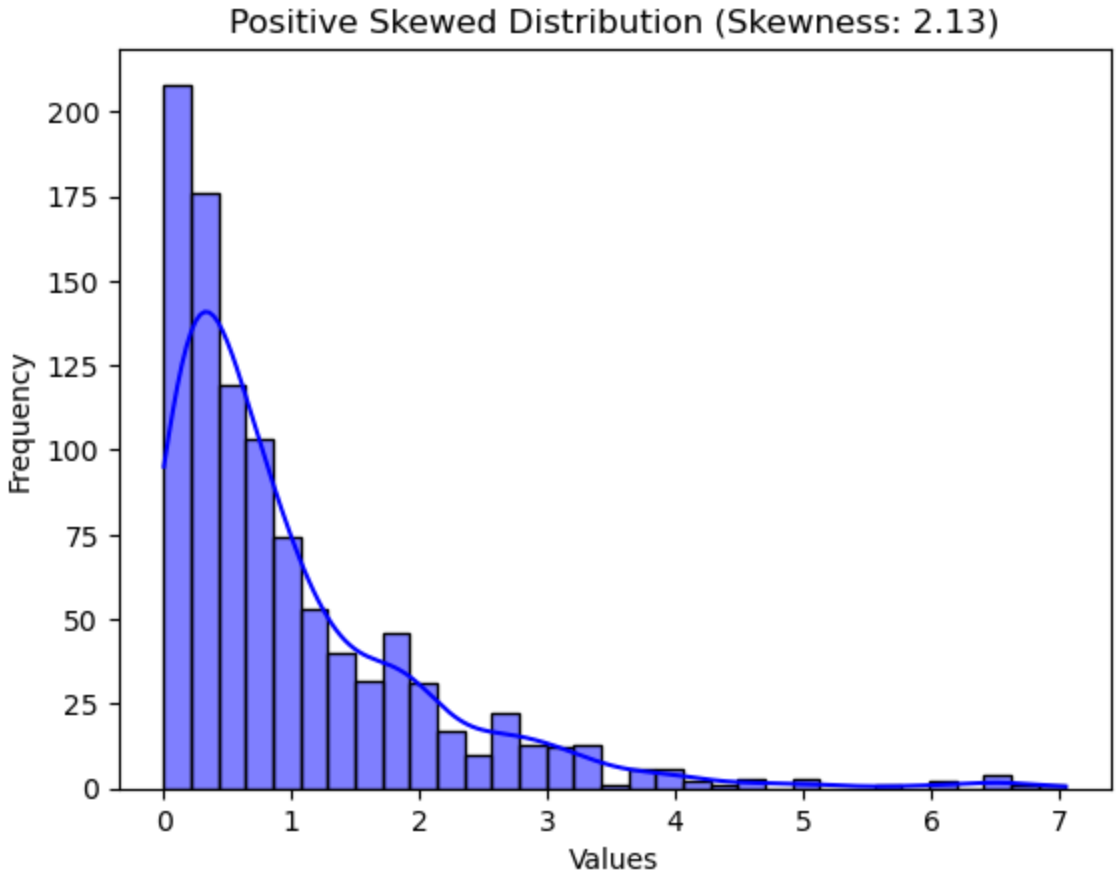
Example:

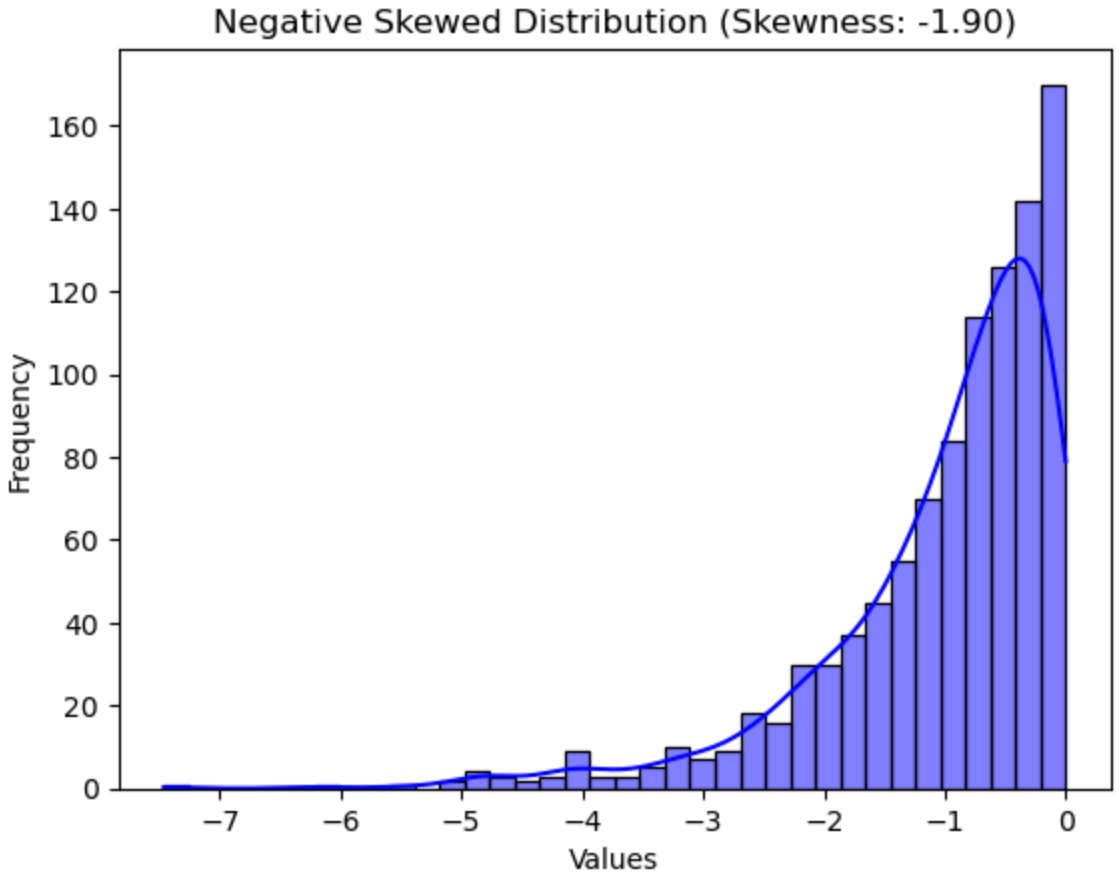
- In a positively skewed distribution, the majority of the values are concentrated on the left side, and the tail extends to the right.

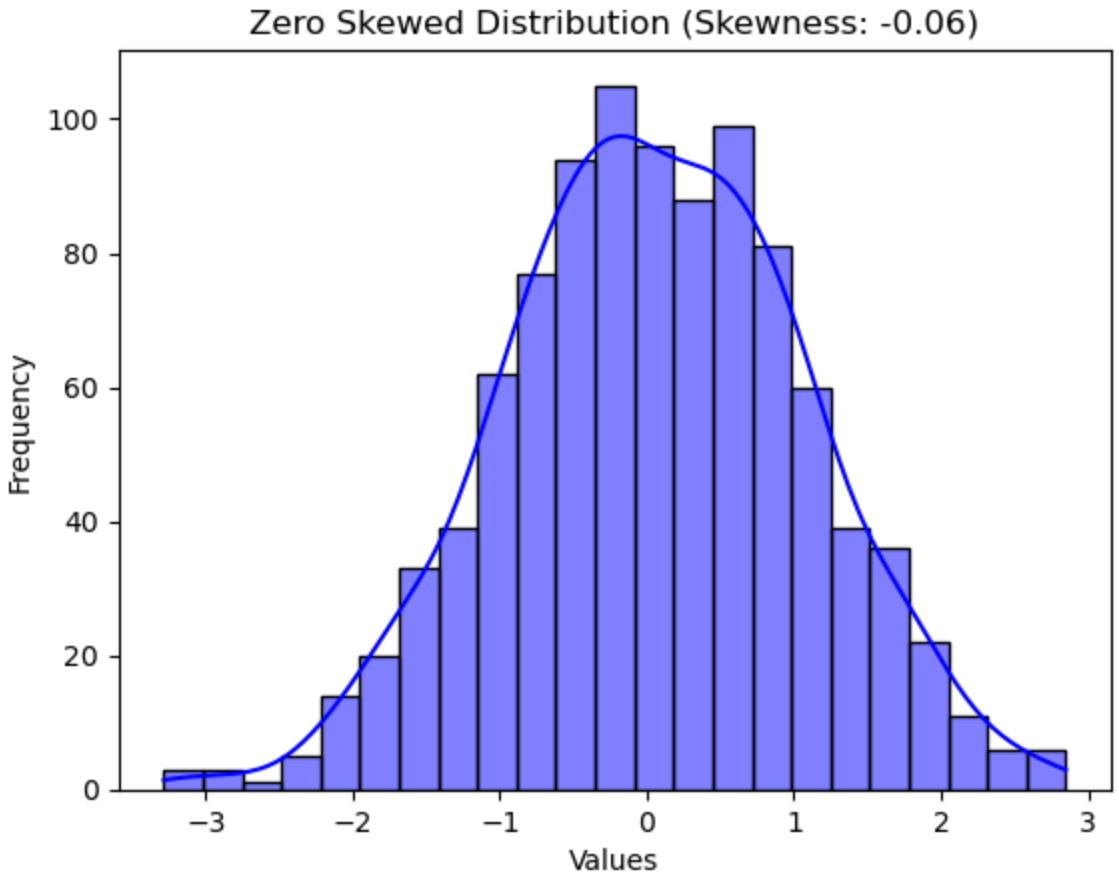
In [20]:

```
import numpy as np
import seaborn as sns
from scipy.stats import skew
import matplotlib.pyplot as plt

# Function to generate skewed datasets and visualize them
def visualize_skewness(skew_type, size=1000):
    if skew_type == 'positive':
        data = np.random.exponential(size=size)
    elif skew_type == 'negative':
        data = -np.random.exponential(size=size)
    else:  # 'zero' for no skewness
        data = np.random.normal(size=size)

    # Calculate skewness
    skewness = skew(data)

    # Visualize the distribution using seaborn
    sns.histplot(data, kde=True, color='blue')
    plt.title(f'{skew_type.capitalize()} Skewed Distribution (Skewness: {skewness:.2f}')
    plt.xlabel('Values')
    plt.ylabel('Frequency')
    plt.show()

# Visualize distributions with different skewness types
visualize_skewness('positive')
```

```
visualize_skewness('negative')
visualize_skewness('zero')
```







In [ ]:

Central Limit Theorem (CLT)

- The central limit theorem states that whenever a random sample of size n is taken from any distribution with mean and variance, then the sample mean will be approximately a normal distribution with mean and variance. The larger the value of the sample size, the better the approximation of the normal.

Assumptions of the Central Limit Theorem

- The sample should be drawn randomly following the condition of randomisation.
- The samples drawn should be independent of each other. They should not influence the other samples.
- When the sampling is done without replacement, the sample size shouldn't exceed 10% of the total population.
- The sample size should be sufficiently large.

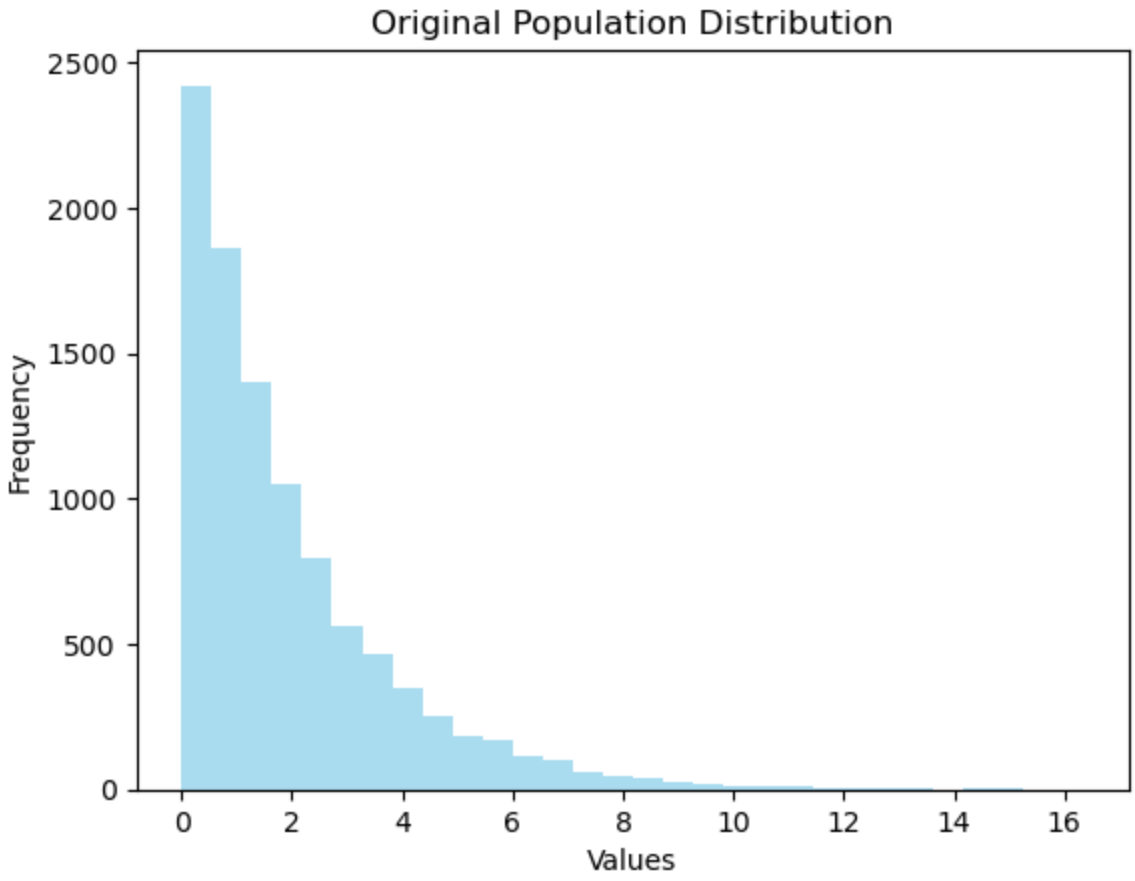
Formula

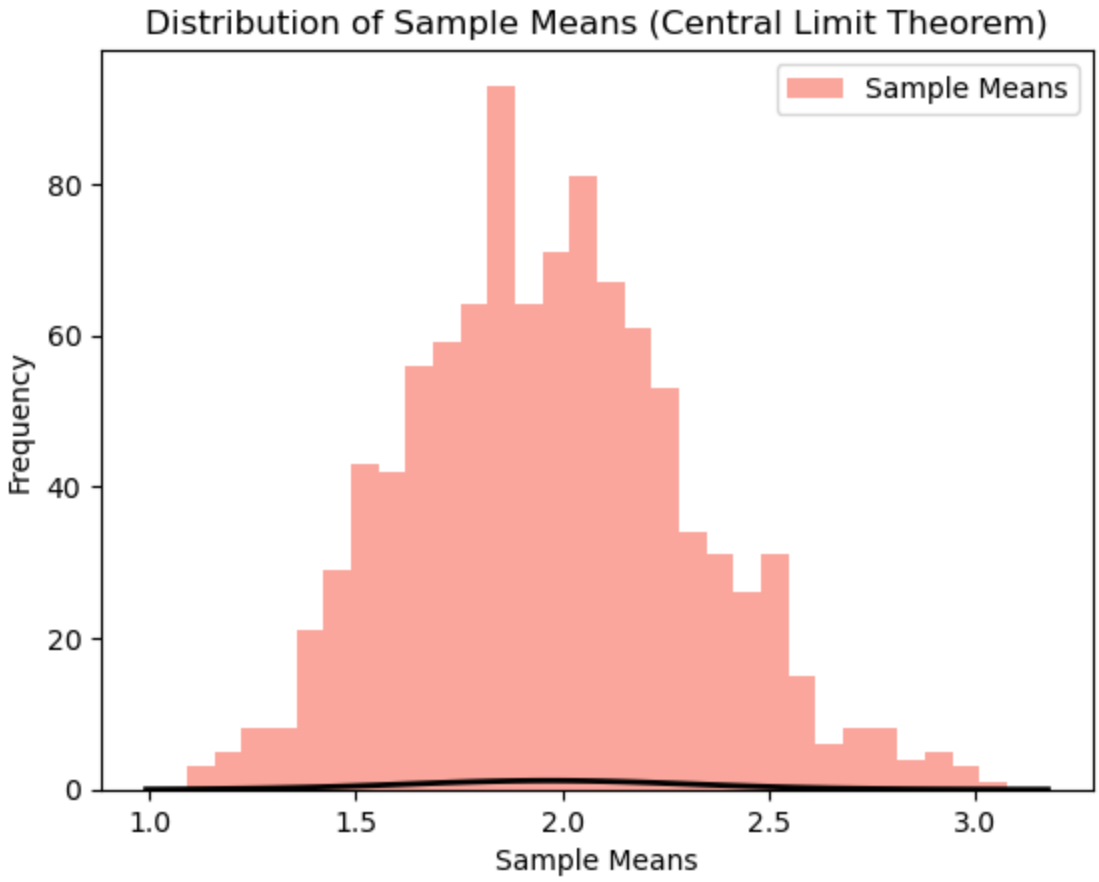
The formula for the central limit theorem is given below:

Central Limit Theorem for Sample Means,

$$Z = \frac{\bar{x} - \mu}{\frac{\sigma}{\sqrt{n}}}$$

In [20]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

# Set a random seed for reproducibility
np.random.seed(42)

# Generate a non-normally distributed population
population = np.random.exponential(scale=2, size=10000)

# Function to calculate sample means
def calculate_sample_means(population, sample_size, num_samples):
    sample_means = []
    for _ in range(num_samples):
        sample = np.random.choice(population, size=sample_size)
        sample_means.append(np.mean(sample))
    return sample_means

# Visualize the original population distribution
plt.hist(population, bins=30, alpha=0.7, color='skyblue')
plt.title('Original Population Distribution')
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.show()

# Visualize the distribution of sample means (CLT)
sample_size = 30
num_samples = 1000

sample_means = calculate_sample_means(population, sample_size, num_samples)

# Fit a normal distribution to the sample means
mu, std = norm.fit(sample_means)

# Plot the histogram of sample means
plt.hist(sample_means, bins=30, alpha=0.7, color='salmon', label='Sample Means')

# Plot the fitted normal distribution
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = norm.pdf(x, mu, std)
plt.plot(x, p, 'k', linewidth=2)

plt.title('Distribution of Sample Means (Central Limit Theorem)')
plt.xlabel('Sample Means')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```





In this example:

- We generate a non-normally distributed population (exponential distribution).
- We visualize the original population distribution.
- We define a function (calculate_sample_means) to simulate the sampling process and calculate sample means.
- We generate a large number of samples and calculate the means.
- We fit a normal distribution to the sample means and visualize the resulting distribution.

Point estimation

- Point estimation is the use of statistics taken from one or several samples to estimate the value of an unknown parameter of a population.

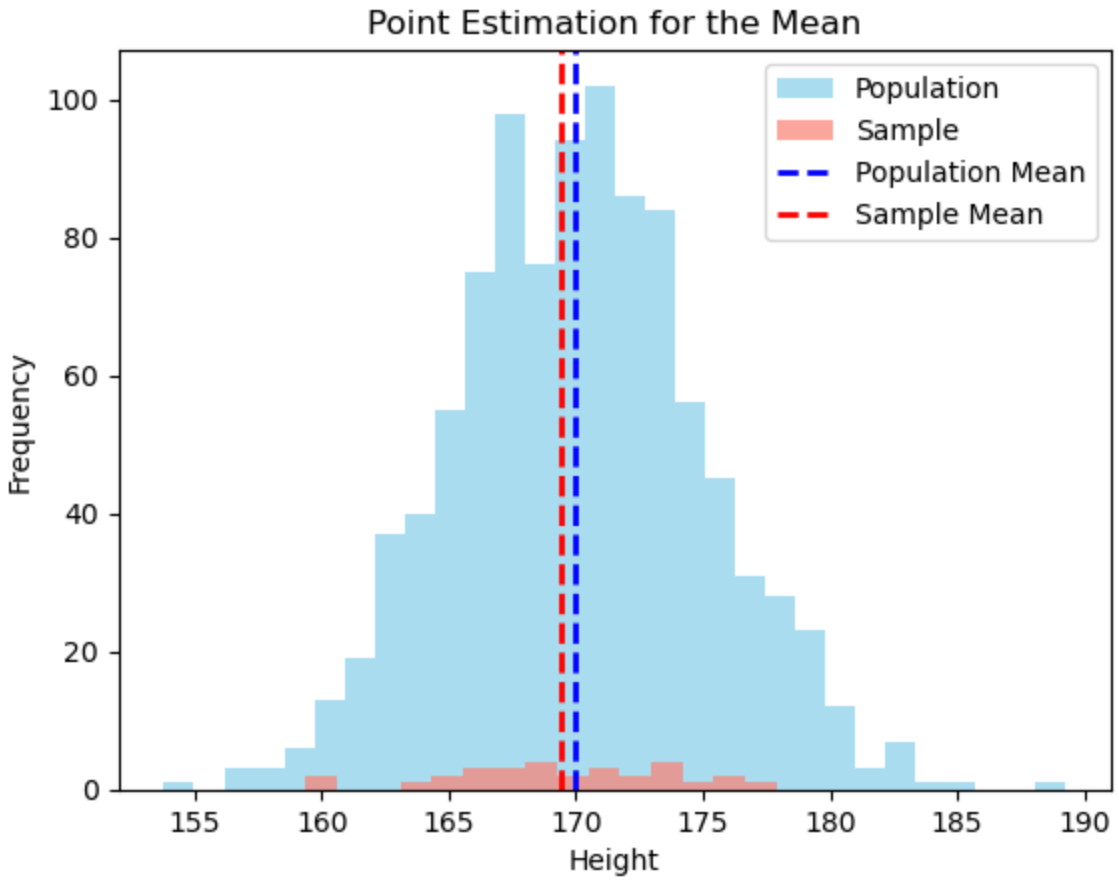
$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n},$$

where

- x_i are the data points (observations) of a sample;
- n is the sample size.

In [21]:
```
#Example: Point Estimation for the Mean
import numpy as np
import matplotlib.pyplot as plt

# Set a random seed for reproducibility
np.random.seed(42)

# Generate a synthetic dataset of heights
population_mean = 170
population_std = 5
sample_size = 30

population = np.random.normal(loc=population_mean, scale=population_std, size=1000)

# Take a random sample from the population
sample = np.random.choice(population, size=sample_size)

# Calculate sample statistics
sample_mean = np.mean(sample)

# Visualize the population distribution and the sample
plt.hist(population, bins=30, alpha=0.7, label='Population', color='skyblue')
plt.hist(sample, bins=15, alpha=0.7, label='Sample', color='salmon')
```

```
plt.axvline(population_mean, color='blue', linestyle='dashed', linewidth=2, label='Pop
plt.axvline(sample_mean, color='red', linestyle='dashed', linewidth=2, label='Sample M
plt.title('Point Estimation for the Mean')
plt.xlabel('Height')
plt.ylabel('Frequency')
plt.legend()
plt.show()

# Display the results
print(f"Sample Mean: {sample_mean}")
```




```
Sample Mean: 169.44996068368962
```

In [ ]:

Hypothesis testing

- Hypothesis testing is a statistical method used to make inferences or decisions about a population based on a sample of data.
- The process involves formulating a hypothesis about the population parameter, collecting and analyzing sample data, and then making a decision about the validity of the hypothesis.

Example

- A teacher assumes that 60% of his college's students come from lower-middle-class families.

Formulate Hypotheses:

Null Hypothesis

- The Null Hypothesis is the assumption that the event will not occur.
- A null hypothesis has no bearing on the study's outcome unless it is rejected.
- H0 is the symbol for it, and it is pronounced H-naught.

Alternate Hypothesis

- The Alternate Hypothesis is the logical opposite of the null hypothesis.
- The acceptance of the alternative hypothesis follows the rejection of the null hypothesis.
- H1 is the symbol for it.

Example

A sanitizer manufacturer claims that its product kills 95 percent of germs on average. To put this company's claim to the test, create a null and alternate hypothesis.

- H0 (Null Hypothesis): Average = 95%.
- Alternative Hypothesis (H1): The average is less than 95%.

P- Value

P-Value or probability value is a number that denotes the likelihood of your data having occurred under the null hypothesis of your statistical test.

When you run the hypothesis test, if you get:

- A small p value (<=0.05), you should reject the null hypothesis
- A large p value (>0.05), you should not reject the null hypothesis

In [ ]:

Types of Hypothesis Testing

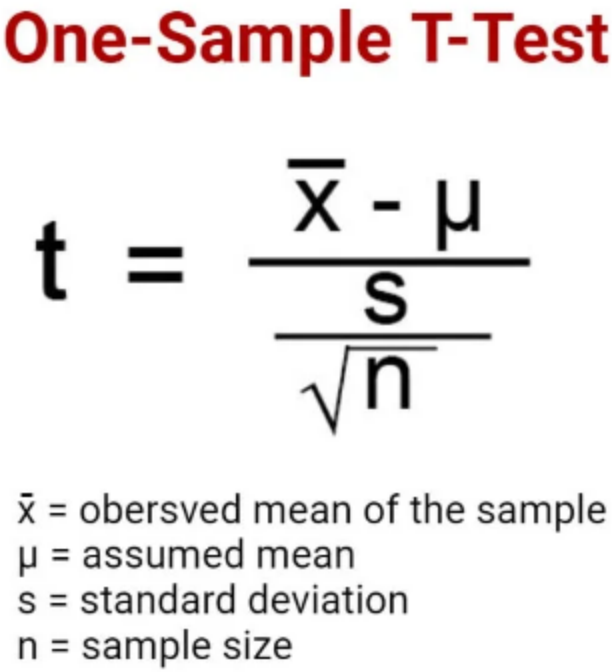
- One-Sample T-Test
- Two-Sample T-Test
- One-Way ANOVA (Analysis of Variance)
- Two-Way ANOVA
- Chi-Square Test

- Z-Test

One-Sample T-Test

- The one-sample t-test is used to determine if the mean of a single sample is significantly different from a known or hypothesized population mean.
- It's commonly applied when you have a sample and you want to test whether the sample mean is statistically different from a specific value.

Formula for One-Sample T-Test:



In [21]:
```
import numpy as np
from scipy.stats import ttest_1samp

# Sample data (random heights in centimeters)
sample_data = np.array([172, 175, 168, 170, 167, 174, 171, 169, 173, 172])

# Hypothesized population mean
population_mean = 170

# Perform one-sample t-test
t_statistic, p_value = ttest_1samp(sample_data, population_mean)

# Output the results
print("Sample Mean:", np.mean(sample_data))
print("Hypothesized Population Mean:", population_mean)
print("T-Statistic:", t_statistic)
print("P-Value:", p_value)

# Interpret the results
alpha = 0.05
if p_value < alpha:
```

```
    print("Reject the null hypothesis")
else:
    print("Fail to reject the null hypothesis")
```

```
Sample Mean: 171.1
Hypothesized Population Mean: 170
T-Statistic: 1.337227482480621
P-Value: 0.2139601131740484
Fail to reject the null hypothesis
```

Two-Sample T-Test

- A two sample t hypothesis tests also known as independent t-test is used to analyze the difference between two unknown population means.
- The Two-sample T-test is used when the two small samples (n< 30) are taken from two different populations and compared.

Assumptions of Two Sample T Hypothesis Tests

- The sample should be randomly selected from the two population
- Samples are independent to each other
- Two sample sizes must me less than 30
- Samples collected from the population are normally distributed

Note:There are (2) types of Two Sample T Hypothesis tests!

Two Sample T Hypothesis Test (Equal Variance)

- Variance of two populations are equal ###### Two Sample T Hypothesis Test (Unequal Variance)
- Variance of two populations are NOT equal

Two Sample T Hypothesis Test (Equal Variance) formula

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

Where:

- $\bar{X}_1$ and $\bar{X}_2$ are the sample means of the two groups.
- s_1^2 and s_2^2 are the sample variances of the two groups.
- n_1 and n_2 are the sample sizes of the two groups.

In [22]:
```
import numpy as np
from scipy import stats

# Generate two random samples (replace this with your actual data)
np.random.seed(42)  # Setting seed for reproducibility
```

```
sample1 = np.random.normal(loc=10, scale=5, size=30)  # Mean=10, Standard Deviation=5,
sample2 = np.random.normal(loc=12, scale=5, size=30)  # Mean=12, Standard Deviation=5,

# Display the generated samples
print("Sample 1:", sample1)
print("Sample 2:", sample2)

# Perform a two-sample t-test
t_statistic, p_value = stats.ttest_ind(sample1, sample2)

# Display the results
print("\nTwo-Sample T-Test Results:")
print("T-Statistic:", t_statistic)
print("P-Value:", p_value)

# Determine statistical significance
alpha = 0.05  # Set your desired significance level
if p_value < alpha:
    print("\nReject the null hypothesis. There is a significant difference between the
else:
    print("\nFail to reject the null hypothesis. There is no significant difference be
```

```
Sample 1: [12.48357077  9.30867849 13.23844269 17.61514928  8.82923313  8.82931522
 17.89606408 13.83717365  7.65262807 12.71280022  7.68291154  7.67135123
 11.20981136  0.43359878  1.37541084  7.18856235  4.9358444  11.57123666
  5.45987962  2.93848149 17.32824384  8.8711185  10.33764102  2.87625907
  7.27808638 10.55461295  4.24503211 11.87849009  6.99680655  8.54153125]
Sample 2: [ 8.99146694 21.26139092 11.93251388  6.71144536 16.11272456  5.89578175
 13.04431798  2.20164938  5.35906976 12.98430618 15.6923329  12.85684141
 11.42175859 10.49448152  4.60739005  8.40077896  9.69680615 17.28561113
 13.71809145  3.18479922 13.62041985 10.0745886   8.61539    15.05838144
 17.15499761 16.6564006   7.80391238 10.45393812 13.65631716 16.87772564]

Two-Sample T-Test Results:
T-Statistic: -1.9751555355027834
P-Value: 0.05301625538101592

Fail to reject the null hypothesis. There is no significant difference between the me
ans.
```

In [ ]:

One-Way ANOVA

- One-Way ANOVA ("analysis of variance") compares the means of two or more independent groups in order to determine whether there is statistical evidence that the associated population means are significantly different.
- One-Way ANOVA is a parametric test.
- This test is also known as: One-Factor ANOVA, One-Way Analysis of Variance

Formula

F-statistics or F-ratio:

F = MSB/MSW

In this formula,

- F = coefficient of ANOVA
- MSB = Mean sum of squares between the groups
- MSW = Mean sum of squares within groups

In [23]:
```
import scipy.stats as stats

# Generate three random samples (replace this with your actual data)
np.random.seed(42)
group1 = np.random.normal(loc=10, scale=5, size=30)
group2 = np.random.normal(loc=12, scale=5, size=30)
group3 = np.random.normal(loc=15, scale=5, size=30)

# Perform One-Way ANOVA
f_statistic, p_value = stats.f_oneway(group1, group2, group3)

# Display the results
print("One-Way ANOVA Results:")
print("F-Statistic:", f_statistic)
print("P-Value:", p_value)

# Determine statistical significance
alpha = 0.05
if p_value < alpha:
    print("\nReject the null hypothesis. At least one group mean is different.")
else:
    print("\nFail to reject the null hypothesis. No significant difference in group me
```

```
One-Way ANOVA Results:
F-Statistic: 12.397871901825155
P-Value: 1.8305780660638664e-05

Reject the null hypothesis. At least one group mean is different.
```

In [ ]:
```

```

Two-Way ANOVA

- Two-Way Analysis of Variance (ANOVA) is a statistical method used to analyze the differences between two independent variables (factors) on a dependent variable.

Assumptions

- The populations from which the samples were obtained must be normally or approximately normally distributed.
- The samples must be independent.
- The variances of the populations must be equal.
- The groups must have the same sample size.

The general formula for a Two-Way ANOVA is as follows:

$$Y_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \epsilon_{ijk}$$

where:

- Y_{ijk} is the observation in the ith level of factor A, jth level of factor B, and kth replicate.
- μ is the overall mean.
- α_i is the effect of the ith level of factor A.
- β_j is the effect of the jth level of factor B.
- $(\alpha\beta)_{ij}$ is the interaction effect between the ith level of factor A and the jth level of factor B.
- ϵ_{ijk} is the random error term.

In [ ]:

In [5]:
```
#!pip install statsmodels
import numpy as np
import pandas as pd
import statsmodels.api as sm
from statsmodels.formula.api import ols

# Set seed for reproducibility
np.random.seed(42)

# Create a DataFrame with random data
data = {'Treatment': np.repeat(['A', 'B', 'C'], 20),
        'Dosage': np.tile(['Low', 'Medium', 'High'], 20),
        'Response': np.random.normal(loc=50, scale=10, size=60)}

df = pd.DataFrame(data)

# Perform Two-Way ANOVA
formula = 'Response ~ C(Treatment) + C(Dosage) + C(Treatment):C(Dosage)'
model = ols(formula, data=df).fit()
anova_table = sm.stats.anova_lm(model, typ=2)

# Print the ANOVA table
print(anova_table)
```

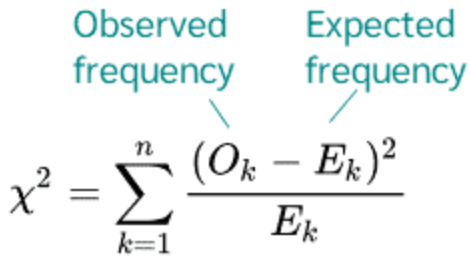
```
                            sum_sq    df         F    PR(>F)
C(Treatment)            66.767723   2.0  0.399935  0.672448
C(Dosage)              141.308604   2.0  0.846430  0.434880
C(Treatment):C(Dosage) 413.389232   4.0  1.238089  0.306542
Residual              4257.136619  51.0       NaN       NaN
```

In [ ]:

Chi-Square Test

- The chi-square test is a hypothesis test used for categorical variables with nominal or ordinal measurement scale.
- The Chi-Square test is a statistical procedure for determining the difference between observed and expected data.

The chi-squared value is calculated via:



In [7]:

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

# Create a contingency table (observed frequencies)
observed = np.array([[30, 10],
                     [15, 25]])

# Perform Chi-Square Test
chi2_stat, p_value, dof, expected = chi2_contingency(observed)

# Print the results
print(f"Chi-Square Statistic: {chi2_stat}")
print(f"P-value: {p_value}")
print(f"Degrees of Freedom: {dof}")
print("Expected Frequencies:")
print(expected)
```

```
Chi-Square Statistic: 9.955555555555556
P-value: 0.0016036472624141077
Degrees of Freedom: 1
Expected Frequencies:
[[22.5 17.5]
 [22.5 17.5]]
```

Z-test

- A Z-test is a statistical test used to determine if there's a significant difference between sample and population means or between the means of two independent samples.
- The test is based on the standard normal distribution (Z distribution) and is commonly used when the sample size is large.

Z-test is Divided into 2 types

- One-Sample Z-Test
- Two-Sample Z-Test

One-Sample Z-Test:

Formula:

$$Z = \frac{\bar{X} - \mu}{\frac{\sigma}{\sqrt{n}}}$$

where:

- Z is the Z-score,
- $\bar{X}$ is the sample mean,
- μ is the population mean (hypothesized value),
- σ is the population standard deviation,
- n is the sample size.

Two-Sample Z-Test (for comparing means of two independent samples):

Formula:

$$Z = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

where:

- Z is the Z-score,
- $\bar{X}_1$ and $\bar{X}_2$ are the means of the two samples,
- s_1 and s_2 are the standard deviations of the two samples,
- n_1 and n_2 are the sample sizes.

Python Example (One-Sample Z-Test):

In [8]:
```
import numpy as np
from scipy.stats import norm

# Generate some random data for illustration
np.random.seed(42)
sample_data = np.random.normal(loc=20, scale=5, size=30)

# Define hypothesized population mean
population_mean = 18

# Perform one-sample Z-test
z_score = (np.mean(sample_data) - population_mean) / (np.std(sample_data) / np.sqrt(le
p_value = 2 * (1 - norm.cdf(np.abs(z_score)))

# Print the results
print(f"Z-Score: {z_score}")
print(f"P-value: {p_value}")
```

```
Z-Score: 1.3113284290316518
P-value: 0.1897468275916272
```

Python Example (Two-Sample Z-Test):

In [10]:
```
import numpy as np
from scipy.stats import norm

# Generate two random samples for illustration
np.random.seed(42)
sample1 = np.random.normal(loc=20, scale=5, size=30)
sample2 = np.random.normal(loc=22, scale=5, size=30)

# Perform two-sample Z-test
def two_sample_z_test(sample1, sample2):
    mean1, mean2 = np.mean(sample1), np.mean(sample2)
    std1, std2 = np.std(sample1), np.std(sample2)
    n1, n2 = len(sample1), len(sample2)

    # Calculate the Z-score
    z_score = (mean1 - mean2) / np.sqrt((std1**2 / n1) + (std2**2 / n2))

    # Two-tailed test, so multiply p-value by 2
    p_value = 2 * (1 - norm.cdf(np.abs(z_score)))

    return z_score, p_value

# Perform the test
z_score, p_value = two_sample_z_test(sample1, sample2)

# Print the results
print(f"Z-Score: {z_score}")
print(f"P-value: {p_value}")
```

```
Z-Score: -2.008921323898585
P-value: 0.044545478101913005
```

In [ ]: