UNIVARIATE:

In data science, **univariate** literally means "**one variable**."

It is a term used to describe a type of data or analysis that focuses on observations involving only a single characteristic or attribute.

Univariate Data

Univariate data is a dataset where each data point is measured or observed for just one variable.

- **Example:** A dataset containing only the **ages** of customers.
    - *The data set is: [25, 31, 45, 22, 50, 31]*
    - Here, 'Age' is the single variable being considered.

Univariate Analysis

Univariate analysis is the simplest form of data analysis. Its main purpose is to **describe** the data and find patterns within that single variable. It is a fundamental part of Exploratory Data Analysis (EDA).

It **doesn't** deal with relationships between variables (like cause and effect or correlation), as only one variable is present.

In the following, each column is a single variable, and they are to be considered individually.

```
In [1]: import pandas as pd
```

```
In [2]: dataset=pd.read_csv("Placement.csv")
```

```
In [3]: dataset
```

Out[3]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary
0	1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	58.80	Placed	270000.0
1	2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Placed	200000.0
2	3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	57.80	Placed	250000.0
3	4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.0	Mkt&HR	59.43	Not Placed	NaN
4	5	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	55.50	Placed	425000.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210	211	M	80.60	Others	82.00	Others	Commerce	77.60	Comm&Mgmt	No	91.0	Mkt&Fin	74.49	Placed	400000.0
211	212	M	58.00	Others	60.00	Others	Science	72.00	Sci&Tech	No	74.0	Mkt&Fin	53.62	Placed	275000.0
212	213	M	67.00	Others	67.00	Others	Commerce	73.00	Comm&Mgmt	Yes	59.0	Mkt&Fin	69.72	Placed	295000.0
213	214	F	74.00	Others	66.00	Others	Commerce	58.00	Comm&Mgmt	No	70.0	Mkt&HR	60.23	Placed	204000.0
214	215	M	62.00	Central	58.00	Others	Science	53.00	Comm&Mgmt	No	89.0	Mkt&HR	60.22	Not Placed	NaN

215 rows × 15 columns

In this session, we are going to consider only numerical data like “ssc_p”, “hsc_p”, etc. Hence, we should separate the quantity type columns from categorical columns.

The following code will do:

```
In [5]: quan = []
        qual = []
        for columnName in dataset.columns:
            #print (columnName)
            if (dataset[columnName].dtype == 'O'):
                #print('qual')
                qual.append(columnName)
            else:
                #print('quan')
                quan.append(columnName)
```

```
In [6]: qual
```

```
Out[6]: ['gender',
         'ssc_b',
         'hsc_b',
         'hsc_s',
         'degree_t',
         'workex',
         'specialisation',
         'status']
```

```
In [7]: quan
```

```
Out[7]: ['sl_no', 'ssc_p', 'hsc_p', 'degree_p', 'etest_p', 'mba_p', 'salary']
```

Central Tendency:

Central tendency is a statistical measure that identifies a single value as representative of an entire dataset or distribution. It aims to find the "centre" or "most typical" value in the data.

It's one of the fundamental concepts in **descriptive statistics**, used to summarize a large set of data with a single, meaningful number.

The Three Main Measures of Central Tendency

The three most common measures are the **mean, median, and mode**. Choosing the right measure depends on the type of data you have and the shape of its distribution.

Measure	Definition	How to Find It	Best Use Case
Mean (Average)	The sum of all values divided by the number of values.	Sum of all data points (∑x) / Count of data points (n)	Best for **symmetrical data** (like a normal distribution) and **interval/ratio data**.
Median	The **middle value** when the data is arranged in order.	Order the data; it's the single middle number (odd n) or the average of the two middle numbers (even n).	Best for **skewed data** or data with **outliers**, as it's less affected by extreme values.

Measure	Definition	How to Find It	Best Use Case
Mode	The value that **occurs most frequently** in the dataset.	Count the frequency of each value and select the most common one.	The **only** measure that can be used for **nominal (categorical) data** (e.g., favourite colour).

Example Scenarios

Data Set	Mean	Median	Mode
A (Symmetric): 2, 4, 6, 8, 10	6.0	6	None (or all)
B (Outlier/Skewed): 2, 4, 6, 8, 100	24.0	6	None (or all)
C (Categorical): Red, Blue, Blue, Green	N/A	N/A	Blue

- **Scenario B** illustrates why the median is important: the single outlier (100) pulls the **mean** up to 24.0, which is not a good representation of the typical value. The **median** of 6 remains a better indicator of the centre.
- **Scenario C** shows the mode is essential for non-numerical data where mean and median are impossible to calculate.

For our considered data set, we used the following code:

```
In [19]: descriptive = pd.DataFrame(index=["mean", "median","mode"], columns = quan)
         for columnName in quan:
             descriptive[columnName]["mean"] = dataset[columnName].mean()
             descriptive[columnName]["median"]= dataset[columnName].median()
             descriptive[columnName]["mode"]= dataset[columnName].mode()[0]
```

```
In [20]: descriptive
```

Out[20]:

	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
mean	108	67.3034	66.3332	66.3702	72.1006	62.2782	288655
median	108	67	65	66	71	62	265000
mode	1	62	63	65	60	56.7	300000

Percentile:

A **percentile** is a statistical measure that indicates the value below which a given **percentage** of observations in a group of observations falls.

In simpler terms, it tells you the **relative standing** of a specific data point within a dataset.

Key Definition and Interpretation

The k-th percentile (Pk) is the value such that k percent of the data is less than or equal to that value.

- If a score is at the **90th percentile**, it means that 90% of the scores in the dataset are less than or equal to that score, and only 10% are greater.
- The **50th percentile** is the **median** (the middle value) of the data set, meaning 50% of the data falls below it.
- The **25th percentile** is also known as the **first quartile** (Q1).
- The **75th percentile** is also known as the **third quartile** (Q3).

Example

Imagine a class of students takes an exam, and the teacher tells a student they scored a **75** on the test, which is in the **85th percentile**.

- **Interpretation:** The score of 75 is greater than or equal to 85% of all other scores in the class. Only 15% of the students scored higher.

Practical Use in Data Science

Percentiles are widely used because they provide a better sense of data distribution than just the mean (average), especially when the data is skewed or contains extreme outliers.

- **Standardized Testing:** To rank students relative to all other test-takers.
- **Health:** To track a child's growth (weight or height) compared to other children their age.
- **IT/Networking:** To measure performance (e.g., the 99th percentile for network latency, which indicates the worst-case performance experienced by 1% of users).

In [47]:
```
descriptive=pd.DataFrame(index=["Mean","Median","Mode","Min","25%","50%","75%","100%"],columns=quan)
for columnName in quan:
    descriptive[columnName]["Mean"]=dataset[columnName].mean()
    descriptive[columnName]["Median"]=dataset[columnName].median()
    descriptive[columnName]["Mode"]=dataset[columnName].mode()[0]
    descriptive[columnName]["Min"]=dataset[columnName].min()
    descriptive[columnName]["25%"]=dataset.describe()[columnName]["25%"]
    descriptive[columnName]["50%"]=dataset.describe()[columnName]["50%"]
    descriptive[columnName]["75%"]=dataset.describe()[columnName]["75%"]
    descriptive[columnName]["100%"]=dataset.describe()[columnName]["max"]
```

In [48]:
```
descriptive
```

Out[48]:

	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
Mean	108	67.3034	66.3332	66.3702	72.1006	62.2782	288655
Median	108	67	65	66	71	62	265000
Mode	1	62	63	65	60	56.7	300000
Min	1	40.89	37	50	50	51.21	200000
25%	54.5	60.6	60.9	61	60	57.945	240000
50%	108	67	65	66	71	62	265000
75%	161.5	75.7	73	72	83.5	66.255	300000
100%	215	89.4	97.7	91	98	77.89	940000

Inter Quartile Range [IQR]

The **Interquartile Range (IQR)** is a measure of statistical dispersion, which describes the spread of the middle half of a dataset. It is defined as the difference between the **third quartile (Q3)** and the **first quartile (Q1)**.

IQR=Q3−Q1

- **Q1 (First Quartile):** The value below which **25%** of the data falls (the 25th percentile).
- **Q3 (Third Quartile):** The value below which **75%** of the data falls (the 75th percentile).

The IQR is a particularly useful measure of variability because it is **robust to outliers**, as it only considers the central 50% of the data, excluding the extreme high and low values. It's often used with the **median** (the second quartile, Q2) to summarize the central tendency and spread of **skewed distributions** or datasets with outliers.

How to Calculate the IQR

To find the IQR for a set of data, follow these steps:

1. **Order the Data:** Arrange the data points from least to greatest.
2. **Find the Median (Q2):** Locate the middle value of the entire dataset. This divides the data into a lower half and an upper half.
3. **Find Q1 (Lower Quartile):** Find the median of the **lower half** of the data.
4. **Find Q3 (Upper Quartile):** Find the median of the **upper half** of the data.
5. **Calculate the IQR:** Subtract Q1 from Q3: IQR=Q3−Q1.

```
In [58]: descriptive=pd.DataFrame(index=["Mean","Median","Mode","Min","25%","50%","75%","100%","IQR",
                                        "1.5IQR","LesserRange","GreaterRange"],columns=quan)
         for columnName in quan:
             descriptive[columnName]["Mean"]=dataset[columnName].mean()
             descriptive[columnName]["Median"]=dataset[columnName].median()
             descriptive[columnName]["Mode"]=dataset[columnName].mode()[0]
             descriptive[columnName]["Min"]=dataset[columnName].min()
             descriptive[columnName]["25%"]=dataset.describe()[columnName]["25%"]
             descriptive[columnName]["50%"]=dataset.describe()[columnName]["50%"]
             descriptive[columnName]["75%"]=dataset.describe()[columnName]["75%"]
             descriptive[columnName]["100%"]=dataset.describe()[columnName]["max"]
             descriptive[columnName]["IQR"]=descriptive[columnName]["75%"]-descriptive[columnName]["25%"]
             descriptive[columnName]["1.5IQR"]=1.5*descriptive[columnName]["IQR"]
             descriptive[columnName]["LesserRange"]=descriptive[columnName]["25%"]- descriptive[columnName]["1.5IQR"]
             descriptive[columnName]["GreaterRange"]=descriptive[columnName]["75%"]+ descriptive[columnName]["1.5IQR"]
```

```
In [59]: descriptive
```

Out[59]:

	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
Mean	108	67.3034	66.3332	66.3702	72.1006	62.2782	288655
Median	108	67	65	66	71	62	265000
Mode	1	62	63	65	60	56.7	300000
Min	1	40.89	37	50	50	51.21	200000
25%	54.5	60.6	60.9	61	60	57.945	240000
50%	108	67	65	66	71	62	265000
75%	161.5	75.7	73	72	83.5	66.255	300000
100%	215	89.4	97.7	91	98	77.89	940000
IQR	107	15.1	12.1	11	23.5	8.31	60000
1.5IQR	160.5	22.65	18.15	16.5	35.25	12.465	90000
LesserRange	-106	37.95	42.75	44.5	24.75	45.48	150000
GreaterRange	322	98.35	91.15	88.5	118.75	78.72	390000

Why 1.5 times of IQR?

The 1.5×IQR rule is a **convention** (a rule of thumb) for defining the "fences" of a box plot, specifically chosen by statistician **John Tukey** as a reasonable boundary for identifying potential outliers.

The primary reason for using 1.5 is that it provides a good balance for detecting points that are statistically unusual, especially in data that follow a **Normal (Gaussian) distribution**.

Application in Outlier Detection

The IQR is commonly used to identify **outliers**—data points that are significantly different from other observations. Potential outliers are often defined as any values that fall outside the range of:

- **Lower Bound:** Q1−1.5×IQR
- **Upper Bound:** Q3+1.5×IQR

The IQR is also prominently displayed in a **boxplot** (or box-and-whisker plot), where the box itself spans from Q1 to Q3, visually representing the interquartile range.

```
In [63]: lesser=[]
         greater=[]

         for columnName in quan:
             if(descriptive[columnName]["LesserRange"]>descriptive[columnName]["Min"]):
                 lesser.append(columnName)
             if(descriptive[columnName]["GreaterRange"]<descriptive[columnName]["100%"]):
                 greater.append(columnName)

In [64]: lesser
Out[64]: ['hsc_p']

In [65]: greater
Out[65]: ['hsc_p', 'degree_p', 'salary']
```

Replacing outliers

Replacing outliers is a common data cleaning step to mitigate their distorting influence on statistical analyses and machine learning models. The choice of replacement method, often called **imputation** or **capping**, depends on the nature of the data, the suspected cause of the outlier, and the goals of the analysis.

Here are the primary methods for replacing (treating) outliers:

1. Winsorization (Capping)

Winsorization (or **capping**) is the most common technique for replacing outliers. It involves setting the extreme values (**outliers**) to a specified percentile value from the main body of the data. This technique reduces the influence of the extreme values without removing the data points entirely, thus preserving the sample size.

- **How it works:**
    - Set a lower and upper percentile threshold (e.g., the 5th and 95th percentiles).
    - All data points **below** the lower threshold are replaced with the value of the **lower threshold**.
    - All data points **above** the upper threshold are replaced with the value of the **upper threshold**.
- **When to use it:** When you believe the extreme values are genuine but want to limit their disproportionate influence on statistics like the mean or on model training. It is useful in applications where keeping the total number of observations is important.

---

2. Imputation with Central Tendency

This method replaces the outlier with a robust measure of central tendency calculated from the non-outlying data.

- **Median Imputation:** Replacing the outlier with the **median** of the non-outlying data.
    - **Advantage:** The median is resistant to outliers, making it a reliable central value. This is generally preferred over the mean when dealing with skewed data or data known to have outliers.
- **Mean Imputation:** Replacing the outlier with the **mean** of the non-outlying data.
    - **Caution:** The mean is sensitive to outliers, so it's generally only suitable if you suspect the original outlier was a simple data-entry error and the data is close to normally distributed.

---

3. Model-Based Imputation

In more sophisticated scenarios, you can use statistical or machine learning models to predict a more plausible value for the outlier.

- **K-Nearest Neighbors (KNN) Imputation:** Replaces the outlier with a weighted average of its K nearest neighbors, using a distance metric calculated from other features.
- **Regression Imputation:** If you have other variables that correlate with the variable containing the outlier, you can build a regression model to predict the outlier value based on the other non-outlying values in that observation.

Alternative: Trimming

While not a replacement method, **trimming** (or **truncation**) is an alternative way to handle outliers by **removing** the extreme data points entirely, leading to a smaller, "trimmed" dataset.

- **How it works:** A certain percentage of the lowest and highest values are deleted from the dataset.
- **When to use it:** When you are confident the outliers are the result of **erroneous data** (e.g., data entry mistakes, malfunctioning sensor) and removing them will not significantly reduce the sample size or introduce bias. If the outlier represents a genuine but rare event, removing it discards valuable information.

In all cases, you should first attempt to **investigate the cause** of the outlier. If an outlier is a clear **measurement or data entry error**, it should be corrected or removed. If it represents a **natural extreme** in the population, a treatment method like Winsorization is usually more appropriate.

For our data, we will go with Winsorization, the process of replacing extreme outlier values with a specified upper and lower limit – capping the outliers.

```
In [68]: for lesscolumn in lesser:
             dataset[lesscolumn][dataset[lesscolumn]<descriptive[lesscolumn]["LesserRange"]]=descriptive[lesscolumn]["LesserRange"]
         for greatcolumn in greater:
             dataset[greatcolumn][dataset[greatcolumn]>descriptive[greatcolumn]["GreaterRange"]]=descriptive[greatcolumn]["GreaterRange"]
```

Action: The code checks if a value in a column (dataset[lesscolumn] or dataset[greatcolumn]) is either less than a defined "LesserRange" or greater than a "GreaterRange".

Replacement: If the condition is true (i.e., the value is an outlier), it replaces the outlier with the boundary value (descriptive[lesscolumn]["LesserRange"] or descriptive[greatcolumn]["GreaterRange"]).

The above code gave a warning message:

SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame

By using .loc, you explicitly tell pandas to select the rows where the condition is true and the specified column (lesscolumn), and then perform the assignment in a single operation, which prevents the ambiguity and suppresses the warning.

```
In [15]: for lesscolumn in lesser:
             dataset.loc [dataset[lesscolumn]<descriptive[lesscolumn]['lesserRange']]=descriptive[lesscolumn]['lesserRange']

         for greatercolumn in greater:
             dataset.loc [dataset[greatercolumn]>descriptive[greatercolumn]['greaterRange']]=descriptive[greatercolumn]['greaterRange']
```