# TASK-3

## Descriptive Statistics

- Descriptive statistics in Python, facilitated by libraries like NumPy, pandas, and Scipy, provide essential tools for summarizing and understanding datasets. With NumPy, users can calculate measures of central tendency, such as **mean**, **median**, and **mode**, as well as measures of variability, like **variance** and **standard deviation,** efficiently. pandas extends this functionality by offering high-level data structures and functions tailored for working with structured data, making it easy to compute summary statistics and perform data aggregation. Additionally, Scipy provides functions for generating descriptive statistics for continuous random variables and working with probability distributions. Through visualization libraries like seaborn and Matplotlib, users can create informative plots such as histograms, box plots, and scatter plots to gain insights into data distributions and relationships. Together, these tools enable Python users to explore, analyze, and interpret data effectively, supporting informed decision-making and further statistical analysis.

# Measures of Central Tendency

- In Python, measures of central tendency provide valuable insights into the typical or central value of a dataset. Here's the detaisl on how to compute each measure using Python libraries:
- **Mean:** The mean, or arithmetic average, is computed by summing all values in the dataset and dividing by the total number of values. It's widely used and easily computed using the np.mean() function in NumPy or the .mean() method in pandas. The mean is sensitive to outliers and can be affected by skewed distributions.
- **Median:** The median represents the middle value of a dataset when arranged in ascending or descending order. It's robust to outliers and skewed distributions, making it preferred in such cases. NumPy's np.median() function and pandas' .median() method efficiently compute the median.
- **Mode**: The mode is the value that appears most frequently in the dataset. It's useful for both numerical and categorical data. While pandas doesn't have a built-in mode method, you can use value_counts() to find the mode(s) with the highest count. Alternatively, SciPy's scipy.stats.mode() function directly computes the mode.

These measures help summarize the central tendency of data and are fundamental in data analysis, helping researchers and analysts gain insights into the characteristics of their datasets. By leveraging Python libraries, analysts can efficiently compute and analyze measures of central tendency, aiding in decision-making and further statistical analysis

# Measures of Dispersion or Variability

- Measures of dispersion in Python, computed using libraries like NumPy, provide crucial insights into the spread or variability of data points within a dataset. Here's the detail s on each measure and how to compute them:
- **Range**: The range is the simplest measure of dispersion, representing the difference between the maximum and minimum values in the dataset. It provides a quick overview of the spread of data.
- **Variance**: Variance quantifies the average squared deviation of each data point from the mean. A higher variance indicates greater dispersion.
- **Standard Deviation:** Standard deviation is the square root of the variance and provides a measure of the average distance of data points from the mean. It's often preferred due to its intuitive interpretation and compatibility with the original data scale.
- **Interquartile Range (IQR):** IQR represents the range covered by the middle 50% of the data and is less sensitive to outliers than the range. It's calculated as the difference between the third quartile (Q3) and the first quartile (Q1).

These measures provide valuable insights into the distribution and variability of data points within a dataset, complementing measures of central tendency. By leveraging Python libraries, analysts can efficiently compute and analyze measures of dispersion, aiding in decision-making and further statistical analysis.

# Quartiles

- Quartiles are statistical measures used to divide a dataset into four equal parts, each representing 25% of the data. They provide insights into the spread and distribution of the data, especially in the context of variability and central tendency. In Python, quartiles can be computed using libraries such as NumPy and pandas.
- **NumPy**: Using np.percentile(), you can specify the desired percentile (25th, 50th, or 75th) to calculate the first quartile (Q1), median (Q2), and third quartile (Q3) of the dataset.
- **pandas**: DataFrame and Series objects have a built-in quantile() method. By passing the desired quartile as a fraction (0.25 for Q1, 0.5 for median, and 0.75 for Q3), you can compute quartiles efficiently.

Understanding quartiles is crucial in various statistical analyses, such as constructing box plots, assessing data variability, and identifying potential outliers. By leveraging Python's capabilities to compute quartiles, analysts can gain deeper insights into the distribution and characteristics of their data, facilitating more informed decision-making and analysis processes.