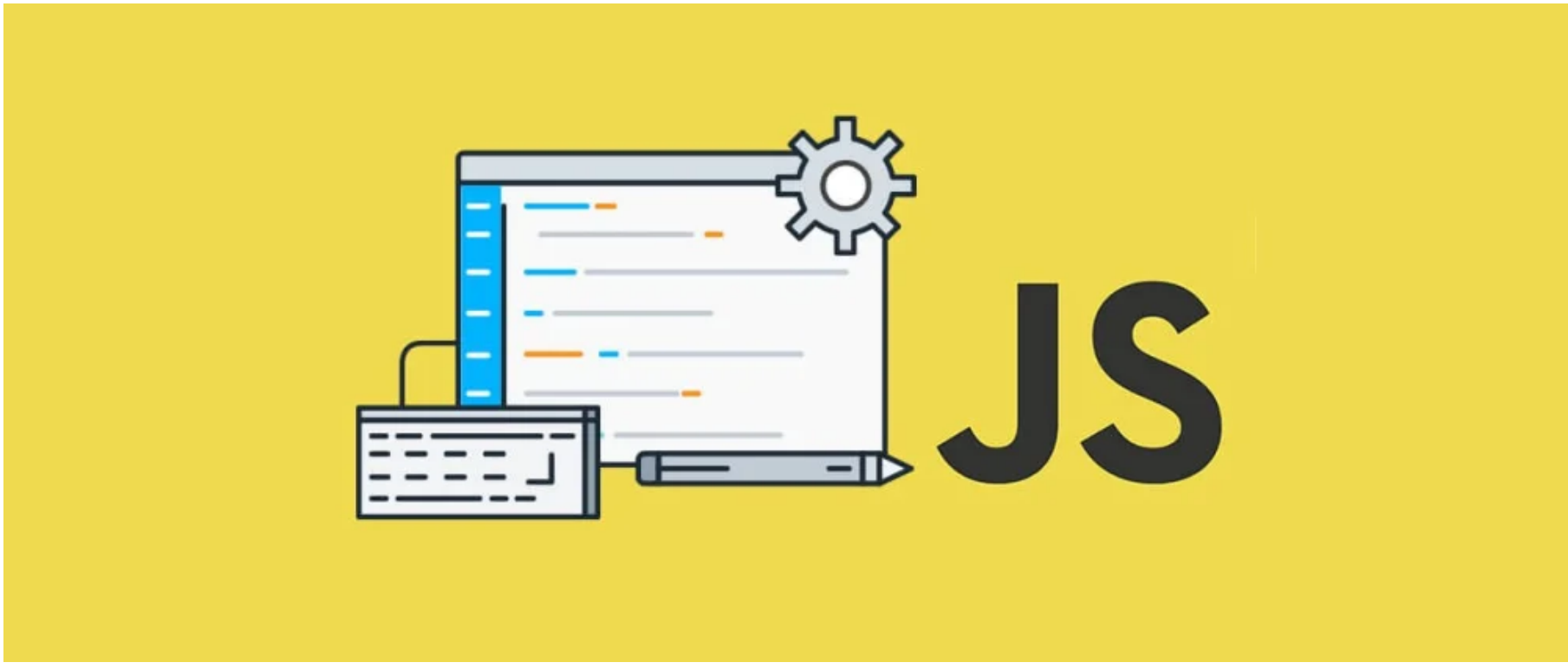
CHAPTER 1 - DESCRIPTIVE STATISTICS (JAVASCRIPT)



Hemant Thapa

1. MEAN

Population Mean

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

Sample Mean

$$\bar{X} = \frac{\sum x}{n}$$

In [8]:
```
function mean(list) {
  let sum = 0;
  for (let i = 0; i < list.length; i++) {
    sum += list[i];
  }
  return sum / list.length;
}
```

In [9]:
```
const numbersList = [4, 8, 15, 16, 23, 42];
const meanValue = mean(numbersList);
console.log('Mean:', meanValue);
```

```
Mean: 18
```

2. MEDIAN

MEDIAN ODD

$$\text{Median} = \left(\frac{n+1}{2}\right)^{th} \text{observation}$$

MEDIAN EVEN

$$\text{Median} = \frac{\frac{n^{th}}{2}\text{obs.} + \left(\frac{n}{2}+1\right)^{th}\text{obs.}}{2}$$

In [10]:
```
function MedianValue(dataset) {
  const sortedDataset = dataset.slice().sort((a, b) => a - b);
  const n = sortedDataset.length;

  let median;
  if (n % 2 === 1) {
    median = sortedDataset[(n - 1) / 2];
  } else {
    const middleRight = n / 2;
    const middleLeft = middleRight - 1;
    median = (sortedDataset[middleLeft] + sortedDataset[middleRight]) / 2;
  }

  console.log("Median:", median);
}
```

In [11]:
```
const numbersDataset = [4, 8, 15, 16, 23, 42];
MedianValue(numbersDataset);
```

```
Median: 15.5
```

3. MODE

Mode = L + (f 1– f 0/2f 1– f 0– f 2) h.

Here h Is the size of class interval

L is the lower limit of the class interval of modal class

f1 is the modal class frequency

f0 is the preceding class frequency

f2 is the succeeding class frequency

In [12]:
```
function modeValue(dataset) {
  let frequencyDict = {};
  let maxFreq = 0;
  let modes = [];

  for (const value of dataset) {
    if (value in frequencyDict) {
      frequencyDict[value]++;
    } else {
      frequencyDict[value] = 1;
    }

    if (frequencyDict[value] > maxFreq) {
      maxFreq = frequencyDict[value];
    }
  }

  for (const [value, frequency] of Object.entries(frequencyDict)) {
    if (frequency === maxFreq) {
      modes.push(parseInt(value)); // Convert value to integer if needed
    }
  }

  console.log("Mode(s):", modes);
}
```

In [13]:
```
const data = [1, 2, 3, 2, 4, 3, 5, 4, 6, 4];
modeValue(data);
```

```
Mode(s): [ 4 ]
```

4. VARIANCE & STANDARD DEVIATION

	Population	Sample
# of subjects	N	n
Mean	$\mu = \frac{\sum_{i=1}^{N} x_i}{N}$	$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$
Variance	$\sigma^2 = \frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N}$	$S^2 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}$

Note: S^2 is the formula for unbiased sample variance, since we're dividing by $n-1$.

Standard deviation	$\sigma = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N}}$	$S = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}}$

Note: Finding S by taking $\sqrt{S^2}$ reintroduces bias.

In [14]:
```javascript
function variance(dataset) {
  const mean = dataset.reduce((acc, curr) => acc + curr, 0) / dataset.length;
  let varianceSum = 0;

  for (const dataPoint of dataset) {
    varianceSum += (dataPoint - mean) ** 2;
  }

  const variance = varianceSum / (dataset.length - 1);
  return variance;
}
```

In [15]:
```javascript
const dataset = [1, 121, 441, 961, 1681, 2601, 3721, 5041, 6561, 8281];
const resultVariance = variance(dataset);

console.log("Dataset:", dataset);
console.log("Variance:", resultVariance);
```

```
Dataset: [
     1,  121,  441,
   961, 1681, 2601,
  3721, 5041, 6561,
  8281
]
Variance: 8345333.333333333
```

In [1]:
```javascript
function calculateStandardDeviation(scores) {
  const mean = scores.reduce((acc, curr) => acc + curr, 0) / scores.length;

  // Calculate deviations
  const deviations = scores.map((x) => x - mean);

  // Calculate square deviations
  const squareDeviations = deviations.map((x) => x ** 2);

  // Create the DataFrame equivalent in JavaScript
  const df_std_dev = scores.map((score, index) => ({
    Score: score,
    Deviation: deviations[index],
    "Square Deviation": squareDeviations[index],
  }));

  // Calculate the total sum of square deviations (standard deviation)
  const standardDeviation = squareDeviations.reduce((acc, curr) => acc + curr, 0);

  return {
    mean,
    deviations,
    squareDeviations,
    df_std_dev,
    standardDeviation,
  };
}
```

In [2]:
```javascript
const score_x = [2, 2, 3, 5];
const result = calculateStandardDeviation(score_x);

console.log("Mean:", result.mean);
console.log("Deviations:", result.deviations);
console.log("Square Deviations:", result.squareDeviations);
console.table(result.df_std_dev);
console.log("Standard Deviation:", result.standardDeviation);
```

Mean: 3
Deviations: [-1, -1, 0, 2]
Square Deviations: [1, 1, 0, 4]

(index)	Score	Deviation	Square Deviation
0	2	-1	1
1	2	-1	1
2	3	0	0
3	5	2	4

Standard Deviation: 6