Calculate summary statistics : Calculate summary statistics (mean, median, mode, standard deviation) for a dataset.

- Dataset : https://www.kaggle.com/c/titanic/data (https://www.kaggle.com/c/titanic/data)

In [2]:
```
import numpy as np
import pandas as pd
```

In [3]:
```
df = pd.read_csv('train.csv')
```

In [6]:
```
#generating descriptive statistics of a DataFrames's numerical columns
df.describe()
```

Out[6]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Choosing the columns to calculate summary statistics.

In [19]:
```
selected_columns = ['Survived','Pclass','Age','SibSp','Parch','Fare']
```

Calculate Summary Statistics

Calculate the mean (average) of the selected columns using Pandas `mean()` function.

In [35]:
```
mean = df[selected_columns].mean()
print(mean)
```

```
Survived     0.383838
Pclass       2.308642
Age         29.699118
SibSp        0.523008
Parch        0.381594
Fare        32.204208
dtype: float64
```

Calculate the median (middle value) of the selected columns using Pandas `median()` function

In [36]:
```
median = df[selected_columns].median()
print(median)
```

```
Survived     0.0000
Pclass       3.0000
Age         28.0000
SibSp        0.0000
Parch        0.0000
Fare        14.4542
dtype: float64
```

Calculate the mode (most frequent value) of the selected columns using Pandas `mode()` function

In [38]:
```
mode = df[selected_columns].mode().iloc[0]
print(mode)
```

```
Survived     0.00
Pclass       3.00
Age         24.00
SibSp        0.00
Parch        0.00
Fare         8.05
Name: 0, dtype: float64
```

Calculate the standard deviation for the selected columns using Pandas `std()` function.

In [37]:
```
std_deviation = df[selected_columns].std()
print(std_deviation)
```

```
Survived     0.486592
Pclass       0.836071
Age         14.526497
SibSp        1.102743
Parch        0.806057
Fare        49.693429
dtype: float64
```

Remove Duplicates: Identify and remove duplicate values in a dataset.

- Dataset : https://www.kaggle.com/datasets/uciml/iris (https://www.kaggle.com/datasets/uciml/iris)

In [2]:
```
import numpy as np
import pandas as pd
```

In [3]:
```
df = pd.read_csv('Iris.csv')
```

In [4]:
```
#displaying first few rows of the DataFrame
df.head()
```

Out[4]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [5]:
```
#displaying last few rows of the DataFrame
df.tail()
```

Out[5]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

In [5]:
```
#getting information about the DataFrame
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [6]:
```
#generating descriptive statistics of a DataFrames's numerical columns
df.describe()
```

Out[6]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [7]:
```
#obtaining a list of columns name in a DataFrame
df.columns
```

Out[7]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidth
Cm',
       'Species'],
      dtype='object')
```

In [8]:
```
#obtaining the data types of each column in a DataFrame
df.dtypes
```

Out[8]:
```
Id                int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

In [9]:
```
#getting the dimensions of a DataFrame
df.shape
```

Out[9]:
```
(150, 6)
```

In [10]:
```
#getting the total number of elements (the count of all the cells) in the D
df.size
```

Out[10]:
```
900
```

Identify Duplicates

In [11]:
```
#identifying duplicate rows
df.duplicated()
```

Out[11]:
```
0      False
1      False
2      False
3      False
4      False
       ...
145    False
146    False
147    False
148    False
149    False
Length: 150, dtype: bool
```

In [13]:
```
df.duplicated(subset = ['SepalLengthCm','SepalWidthCm','PetalLengthCm','Pet
```

Out[13]:
```
3
```

Removing Duplicates

In [14]:
```
#removing duplicates
df.drop_duplicates(subset = ['SepalLengthCm','SepalWidthCm','PetalLengthCm'
```

Check whether duplicates are removed

In [15]:
```
#checking whether duplicates are removed
df.duplicated(subset = ['SepalLengthCm','SepalWidthCm','PetalLengthCm','Pet
```

Out[15]:
```
0
```