LOADING DATASET

In [2]:
```
import pandas as pd
import warnings
warnings.filterwarnings('ignore', category=UserWarning)

#LOad the dataset
df = pd.read_csv('books_data.csv')
```

Data Preview

In [3]:
```
df.head(5)
```

Out[3]:

	Unnamed: 0	Books	Authors	Language	First_Published	Sales_in_millions
0	0	A Tale of Two Cities	Charles Dickens	English	1859	200.0
1	1	The Little Prince (Le Petit Prince)	Antoine de Saint-Exupéry	French	1943	200.0
2	2	Harry Potter and the Philosopher's Stone	J. K. Rowling	English	1997	120.0
3	3	And Then There Were None	Agatha Christie	English	1939	100.0
4	4	Dream of the Red Chamber (紅樓夢)	Cao Xueqin	Chinese	1791	100.0

In [4]:
```
df.tail(5)
```

Out[4]:

	Unnamed: 0	Books	Authors	Language	First_Published	Sales_in_millions
285	285	The No. 1 Ladies Detective Agency	Alexander McCall Smith	English	1999–present	15.0
286	286	Der Regenbogenfisch (Rainbow Fish)	Marcus Pfister	German	1992–present	15.0
287	287	The Riftwar Cycle	Raymond E. Feist	English	1982–present	15.0
288	288	The Thrawn trilogy	Timothy Zahn	English	1991–93	15.0
289	289	Wiedźmin (The Witcher)	Andrzej Sapkowski	Polish	1990–2013	15.0

RENAME A COLUMN

In [5]:
```
df.rename(columns={'Unnamed: 0': 'Sl No.'}, inplace=True)
```

In [6]:
```
df.head()
```

Out[6]:

	Sl No.	Books	Authors	Language	First_Published	Sales_in_millions
0	0	A Tale of Two Cities	Charles Dickens	English	1859	200.0
1	1	The Little Prince (Le Petit Prince)	Antoine de Saint-Exupéry	French	1943	200.0
2	2	Harry Potter and the Philosopher's Stone	J. K. Rowling	English	1997	120.0
3	3	And Then There Were None	Agatha Christie	English	1939	100.0
4	4	Dream of the Red Chamber (紅樓夢)	Cao Xueqin	Chinese	1791	100.0

DATA TYPES

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 290 entries, 0 to 289
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Sl No.             290 non-null    int64  
 1   Books              290 non-null    object 
 2   Authors            290 non-null    object 
 3   Language           290 non-null    object 
 4   First_Published    290 non-null    object 
 5   Sales_in_millions  288 non-null    float64
dtypes: float64(1), int64(1), object(4)
memory usage: 13.7+ KB
```

In [8]:
```
df.dtypes
```

Out[8]:
```
Sl No.                 int64
Books                 object
Authors               object
Language              object
First_Published       object
Sales_in_millions    float64
dtype: object
```

MISSING VALUES

In [9]:
```
df.isna().sum()
```

Out[9]:
```
Sl No.               0
Books                0
Authors              0
Language             0
First_Published      0
Sales_in_millions    2
dtype: int64
```

In [10]:
```
df.isnull().mean()*100
```

Out[10]:
```
Sl No.               0.000000
Books                0.000000
Authors              0.000000
Language             0.000000
First_Published      0.000000
Sales_in_millions    0.689655
dtype: float64
```

In [11]:
```
df.isnull()
```

Out[11]:

	Sl No.	Books	Authors	Language	First_Published	Sales_in_millions
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
...	...	...	...	...	...	...
285	False	False	False	False	False	False
286	False	False	False	False	False	False
287	False	False	False	False	False	False
288	False	False	False	False	False	False
289	False	False	False	False	False	False

290 rows × 6 columns

REMOVING WITH MISSING VALUES:

We can use the dropna() function to remove rows with missing values. However, this approach may lead to loss of valuable data if the missing values are present in important rows.

In [13]:
```
df.dropna(subset=['Sales_in_millions'])
```

Out[13]:

	Sl No.	Books	Authors	Language	First_Published	Sales_in_millions
0	0	A Tale of Two Cities	Charles Dickens	English	1859	200.0
1	1	The Little Prince (Le Petit Prince)	Antoine de Saint-Exupéry	French	1943	200.0
2	2	Harry Potter and the Philosopher's Stone	J. K. Rowling	English	1997	120.0
3	3	And Then There Were None	Agatha Christie	English	1939	100.0
4	4	Dream of the Red Chamber (紅樓夢)	Cao Xueqin	Chinese	1791	100.0
...	...	...	...	...	...	...
285	285	The No. 1 Ladies Detective Agency	Alexander McCall Smith	English	1999–present	15.0
286	286	Der Regenbogenfisch (Rainbow Fish)	Marcus Pfister	German	1992–present	15.0
287	287	The Riftwar Cycle	Raymond E. Feist	English	1982–present	15.0
288	288	The Thrawn trilogy	Timothy Zahn	English	1991–93	15.0
289	289	Wiedźmin (The Witcher)	Andrzej Sapkowski	Polish	1990–2013	15.0

288 rows × 6 columns

Impute Missing Values:

Another approach is to impute the missing values with a suitable value, such as the mean, median, or mode of the column. Since the "Sales_in_millions" column contains float values, imputing with the mean may be a reasonable choice.

In [14]:
```
mean_sales = df['Sales_in_millions'].mean()
df['Sales_in_millions'].fillna(mean_sales, inplace=True)
```

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_12376\3423858422.py:2: FutureWarning: A value is trying to be set o
n a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on w
hich we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)'
or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  df['Sales_in_millions'].fillna(mean_sales, inplace=True)
```

Interpolate Missing Values:

If the missing values are sequential and have some logical progression, you can use interpolation to estimate the missing values based on neighboring data points.

In [15]:
```
df['Sales_in_millions'].interpolate(method='linear', inplace=True)
```

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_12376\1200853332.py:1: FutureWarning: A value is trying to be set o
n a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on w
hich we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)'
or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  df['Sales_in_millions'].interpolate(method='linear', inplace=True)
```

In [16]:
```
df.isna().sum()
```

Out[16]:
```
Sl No.              0
Books               0
Authors             0
Language            0
First_Published     0
Sales_in_millions   0
dtype: int64
```

In [17]:
```
df.isnull().mean()*100
```

Out[17]:
```
Sl No.              0.0
Books               0.0
Authors             0.0
Language            0.0
First_Published     0.0
Sales_in_millions   0.0
dtype: float64
```

STATISTICAL OVERVIEW

In [18]:
```
df.describe().transpose()
```

Out[18]:

	count	mean	std	min	25%	50%	75%	max
Sl No.	290.0	144.500000	83.860002	0.0	72.25	144.5	216.75	289.0
Sales_in_millions	290.0	49.996875	64.460421	10.0	16.00	25.0	50.00	600.0

In [19]:
```
df["Sales_in_millions"].skew()
```

Out[19]: 3.8678079494219357

As you can see I have used "skew()" Function. So in pandas, the skew() function is used to compute the skewness of a numerical column in a DataFrame. Skewness is a measure of the asymmetry of the distribution of values in a dataset. It indicates whether the data is skewed to the left or right of the mean. Positive Skewness (>0): Also known as right-skewed distribution, it means that the tail of the distribution extends towards the right, indicating that the majority of the data points are concentrated on the left side of the distribution, with a few large values extending the tail to the right. In such cases, the mean is typically greater than the median. Negative Skewness (<0): Also known as left-skewed distribution, it means that the tail of the distribution extends towards the left, indicating that the majority of the data points are concentrated on the right side of the distribution, with a few small values extending the tail to the left. In such cases, the mean is typically less than the median. Zero Skewness (0): A skewness value close to zero indicates a symmetric distribution where the data is evenly distributed on both sides of the mean, and the tails of the distribution are balanced.

DUPLICATE DATA

Identifying Duplicates:

In [20]:
```
duplicate_rows = df.duplicated()
```

Viewing Duplicate Rows:

In [21]:
```
duplicate_data = df[duplicate_rows]
```

Counting Duplicate Rows:

In [22]:
```
num_duplicates = duplicate_rows.sum()
print("Total number of duplicate rows:", num_duplicates)
```

```
Total number of duplicate rows: 0
```

If there are any duplicate rows in the dataset, they can be removed from the DataFrame using the 'drop_duplicates()' function. By default, this function retains the first occurrence of each unique row, effectively removing subsequent duplicates. Additionally, if you want to identify duplicates based on specific columns, you can achieve this by specifying the column names using the 'subset' parameter of the 'duplicated()' function. This allows you to target specific columns for duplicate identification and removal, ensuring data integrity and accuracy.

STANDARDIZATION

IMPORTING NECESSARY LIBARIES

In [23]:
```
pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\python310\lib\site-packages (1.4.0)
Requirement already satisfied: joblib>=1.2.0 in c:\python310\lib\site-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\python310\lib\site-packages (from scikit-learn) (3.2.0
)
Requirement already satisfied: scipy>=1.6.0 in c:\python310\lib\site-packages (from scikit-learn) (1.12.0)
Requirement already satisfied: numpy<2.0,>=1.19.5 in c:\python310\lib\site-packages (from scikit-learn) (1.26.4)
Note: you may need to restart the kernel to use updated packages.
```

```
WARNING: Ignoring invalid distribution -p (c:\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -p (c:\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\python310\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -p (c:\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -p (c:\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\python310\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -p (c:\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -p (c:\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\python310\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -p (c:\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -p (c:\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\python310\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -p (c:\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -p (c:\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\python310\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -p (c:\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -p (c:\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\python310\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\python310\lib\site-packages)

[notice] A new release of pip available: 22.2.2 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [24]:
```
from sklearn.preprocessing import StandardScaler
```

Convert data to DataFrame

In [25]:
```
df_std = pd.DataFrame(df)
```

Extract the numerical column for standardization

We begin by extracting the column "Sales_in_millions" from the dataset. This column contains numerical data that we want to standardize.

In [26]:
```
sales_column = df_std['Sales_in_millions']
```

Reshape the data for StandardScaler (required for compatibility)

In [27]:
```
sales_data = sales_column.values.reshape(-1, 1)
```

Fit and transform the data for standardization

we calculate the mean (average) and standard deviation of the "Sales_in_millions" column. The mean represents the center of the data, while the standard deviation measures the spread or dispersion of the data points.

Standardization involves transforming the data such that it has a mean of 0 and a standard deviation of 1. To achieve this, we subtract the mean from each value in the column to center the data around 0. Then, we divide each value by the standard deviation to scale the data.

In [28]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
```

In [29]:
```
standardized_sales = scaler.fit_transform(sales_data)
```

Update the DataFrame with the standardized values

After standardizing the data, we update the original dataset by replacing the "Sales_in_millions" column with the standardized values. This ensures that the dataset reflects the standardized version of the numerical feature.

In [30]:
```
df_std['Standardized_Sales'] = standardized_sales
```

In [31]:
```
df_std
```

Out[31]:

	Sl No.	Books	Authors	Language	First_Published	Sales_in_millions	Standardized_Sales
0	0	A Tale of Two Cities	Charles Dickens	English	1859	200.0	2.331080
1	1	The Little Prince (Le Petit Prince)	Antoine de Saint-Exupéry	French	1943	200.0	2.331080
2	2	Harry Potter and the Philosopher's Stone	J. K. Rowling	English	1997	120.0	1.087863
3	3	And Then There Were None	Agatha Christie	English	1939	100.0	0.777059
4	4	Dream of the Red Chamber (紅樓夢)	Cao Xueqin	Chinese	1791	100.0	0.777059
...	...	...	...	...	...	...	...
285	285	The No. 1 Ladies Detective Agency	Alexander McCall Smith	English	1999–present	15.0	-0.543859
286	286	Der Regenbogenfisch (Rainbow Fish)	Marcus Pfister	German	1992–present	15.0	-0.543859
287	287	The Riftwar Cycle	Raymond E. Feist	English	1982–present	15.0	-0.543859
288	288	The Thrawn trilogy	Timothy Zahn	English	1991–93	15.0	-0.543859
289	289	Wiedźmin (The Witcher)	Andrzej Sapkowski	Polish	1990–2013	15.0	-0.543859

290 rows × 7 columns

A DataFrame with the "Sales_in_millions" column replaced by the standardized values in the "Standardized_Sales" column.

NORMALIZATION

Import necessary libraries

In [32]:
```
from sklearn.preprocessing import MinMaxScaler
```

Convert Data To DataFrame

In [33]:
```
df_nrm=pd.DataFrame(df)
```

Extract The Numberical Column For Normalization

We begin by extracting the column "Sales_in_millions" from the dataset. This column contains numerical data that we want to normalize.

In [34]:
```
sales_columns = df_nrm['Sales_in_millions']
```

Initialize The MinMaxScaler

We compute the minimum and maximum values of the "Sales_in_millions" column. These values will be used to scale the data to the desired range.

In [35]:
```
scaler=MinMaxScaler()
```

Reshape The Data For MinMaxScaler (Required For Compatibility)

In [36]:
```
sales_datas = sales_columns.values.reshape(-1, 1)
```

Fit And Transform The Data For Normalization

Normalization involves transforming the data such that it falls within a specified range, often between 0 and 1. To achieve this, we subtract the minimum value from each data point to shift the range to start from 0. Then, we divide each value by the range (i.e., the difference between the maximum and minimum values) to scale the data.

In [37]:
```
normalized_sales = scaler.fit_transform(sales_datas)
```

Update The DataFrame With The Normalized Values

After normalizing the data, we update the original dataset by replacing the "Sales_in_millions" column with the normalized values. This ensures that the dataset reflects the normalized version of the numerical feature.

In [38]:
```
df_nrm['Normalized_Sales'] = normalized_sales
```

In [39]:
```
df_nrm
```

Out[39]:

	Sl No.	Books	Authors	Language	First_Published	Sales_in_millions	Normalized_Sales
0	0	A Tale of Two Cities	Charles Dickens	English	1859	200.0	0.322034
1	1	The Little Prince (Le Petit Prince)	Antoine de Saint-Exupéry	French	1943	200.0	0.322034
2	2	Harry Potter and the Philosopher's Stone	J. K. Rowling	English	1997	120.0	0.186441
3	3	And Then There Were None	Agatha Christie	English	1939	100.0	0.152542
4	4	Dream of the Red Chamber (紅樓夢)	Cao Xueqin	Chinese	1791	100.0	0.152542
...	...	...	...	...	...	...	...
285	285	The No. 1 Ladies Detective Agency	Alexander McCall Smith	English	1999–present	15.0	0.008475
286	286	Der Regenbogenfisch (Rainbow Fish)	Marcus Pfister	German	1992–present	15.0	0.008475
287	287	The Riftwar Cycle	Raymond E. Feist	English	1982–present	15.0	0.008475
288	288	The Thrawn trilogy	Timothy Zahn	English	1991–93	15.0	0.008475
289	289	Wiedźmin (The Witcher)	Andrzej Sapkowski	Polish	1990–2013	15.0	0.008475

290 rows × 7 columns

A DataFrame with the "Sales_in_millions" column replaced by the normalized values in the "Normalized_Sales" column.

In [ ]:
```

```