In [3]:
```
import pandas as pd
```

Create a Dataframe

In [15]:
```
students_stats = {'Name' : ['Tina', 'Aisha', 'Gbemisola', 'Desmond', 'Shenko', 'K
                  'Age' : [30, 29, 25, 32, 27, 35],
                  'Height' : [6,6,6.5,5,5,5.5,]}

SS = pd.DataFrame(students_stats)
```

In [16]:
```
SS
```

Out[16]:

	Name	Age	Height
0	Tina	30	6.0
1	Aisha	29	6.0
2	Gbemisola	25	6.5
3	Desmond	32	5.0
4	Shenko	27	5.0
5	Kingsley	35	5.5

1) The head() function: it is used to select the top 5 data in the dataframe

In [17]:
```
SS.head()
```

Out[17]:

	Name	Age	Height
0	Tina	30	6.0
1	Aisha	29	6.0
2	Gbemisola	25	6.5
3	Desmond	32	5.0
4	Shenko	27	5.0

2) The tail() function: it is used to select the bottom 5 data in the dataframe

In [21]:
```
SS.tail()
```

Out[21]:

	Name	Age	Height
1	Aisha	29	6.0
2	Gbemisola	25	6.5
3	Desmond	32	5.0
4	Shenko	27	5.0
5	Kingsley	35	5.5

3) Data dataframe.info() function is used to get a concise summary of the dataframe

In [23]:
```
SS.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Name    6 non-null      object 
 1   Age     6 non-null      int64  
 2   Height  6 non-null      float64
dtypes: float64(1), int64(1), object(1)
memory usage: 272.0+ bytes
```

4) Dtypes: it is used to show the data type of each column.

In [24]:
```
SS.dtypes
```

Out[24]:
```
Name       object
Age         int64
Height    float64
dtype: object
```

5) Shape and size function

The Shape shows the number of dimensions as well as the size in each dimension, while the Size, as the name suggests, returns the size of a dataframe which is the number of rows multiplied by the number of columns.

In [25]:
```
SS.shape
```

Out[25]: (6, 3)

In [26]:
```
SS.size
```

Out[26]: 18

6) describe( ) function is used to do a quick statistical summary for every numerical column, as shown below

In [27]:
```
SS.describe()
```

Out[27]:

	Age	Height
count	6.000000	6.000000
mean	29.666667	5.666667
std	3.559026	0.605530
min	25.000000	5.000000
25%	27.500000	5.125000
50%	29.500000	5.750000
75%	31.500000	6.000000
max	35.000000	6.500000

7) Sample method: it allows you to select values randomly from a Series or DataFrame. It is useful when we want to select a random sample from a distribution.

In [36]:
```
SS.sample(3)
```

Out[36]:

	Name	Age	Height
5	Kingsley	35	5.5
2	Gbemisola	25	6.5
0	Tina	30	6.0

8) isnull () function: it is used to Identify Missing Values

In [37]:
```
SS.isnull
```

Out[37]:
```
<bound method DataFrame.isnull of         Name  Age  Height
0       Tina   30     6.0
1      Aisha   29     6.0
2  Gbemisola   25     6.5
3    Desmond   32     5.0
4     Shenko   27     5.0
5   Kingsley   35     5.5>
```

9) isna function: it is used to returns a dataframe filled with boolean values with false indicating no missing values. But if there were missing values it will be true

In [39]:
```
SS.isna().any()
```

Out[39]:
```
Name      False
Age       False
Height    False
dtype: bool
```

10) df.isnull().sum() function: it is used to compute the total number of missing values in the data frame

In [44]:
```
SS.isnull().sum()
```

Out[44]:
```
Name      0
Age       0
Height    0
dtype: int64
```

11) Nunique() function: it counts the number of unique entries over columns or rows.

In [45]:
```
SS.nunique()
```

Out[45]:
```
Name      6
Age       6
Height    4
dtype: int64
```

12) Index( ) and column( )

index() is an inbuilt function in Python, which searches for a given element from the start of the list and returns the lowest index where the element appears.

In [53]:
```
SS.index
```

Out[53]:
```
RangeIndex(start=0, stop=6, step=1)
```

In [49]:
```
SS.columns
```

Out[49]:
```
Index(['Name', 'Age', 'Height'], dtype='object')
```

13) Memory_usage(): this returns how much memory each column uses in bytes. It is useful especially when we work with large dataframes.

In [51]:
```
SS.memory_usage()
```

Out[51]:
```
Index     128
Name       48
Age        48
Height     48
dtype: int64
```

14) Loc and iloc

Loc and iloc are used to select rows and columns.

loc: select by labels

iloc: select by positions

In [55]:
```
SS.loc[:3]
```

Out[55]:

	Name	Age	Height
0	Tina	30	6.0
1	Aisha	29	6.0
2	Gbemisola	25	6.5
3	Desmond	32	5.0

In [56]:
```
SS.iloc[:3]
```

Out[56]:

	Name	Age	Height
0	Tina	30	6.0
1	Aisha	29	6.0
2	Gbemisola	25	6.5

15) Slicing

Slicing Rows and Columns using labels. You can select a range of rows or columns using labels or by position.

In [57]:
```
SS[0:4]
```

Out[57]:

	Name	Age	Height
0	Tina	30	6.0
1	Aisha	29	6.0
2	Gbemisola	25	6.5
3	Desmond	32	5.0

16) Groupby

pandas groupby function is a great tool in exploring the data. It makes it easier to unveil the underlying relationships among variables.

In [64]:
```
SS.groupby(['Name']).mean()
```

Out[64]:

Name	Age	Height
Aisha	29.0	6.0
Desmond	32.0	5.0
Gbemisola	25.0	6.5
Kingsley	35.0	5.5
Shenko	27.0	5.0
Tina	30.0	6.0

17) Sorting

Using the sort_index() method, by passing the axis arguments and the order of sorting, DataFrame can be sorted. By default, sorting is done on row labels in ascending order.

In [88]:
```
SS.sort_index()
```

Out[88]:

	Name	Age	Height
0	Tina	30	6.0
1	Aisha	29	6.0
2	Gbemisola	25	6.5
3	Desmond	32	5.0
4	Shenko	27	5.0
5	Kingsley	35	5.5

In [102]:
```
SS.sort_values
```

Out[102]:
```
<bound method DataFrame.sort_values of         Name  Age  Height
0       Tina   30     6.0
1      Aisha   29     6.0
2  Gbemisola   25     6.5
3    Desmond   32     5.0
4     Shenko   27     5.0
5   Kingsley   35     5.5>
```

18) Dropna

The dropna () function is used to remove a row or a column from a dataframe which has a NaN or no values in it.

In [112]:
```
drop_cols = ['Height']

SS.drop(drop_cols,1)
```

```
C:\Users\AMANO JUSTINA\AppData\Local\Temp\ipykernel_11816\64139880.py:3: Future
Warning: In a future version of pandas all arguments of DataFrame.drop except f
or the argument 'labels' will be keyword-only.
  SS.drop(drop_cols,1)
```

Out[112]:

	Name	Age
0	Tina	30
1	Aisha	29
2	Gbemisola	25
3	Desmond	32
4	Shenko	27
5	Kingsley	35

19) Query

We sometimes need to filter a dataframe based on a condition or apply a mask to get certain values. One easy way to filter a dataframe is query function. Let's first create a sample dataframe.

In [116]:
```
SS.query('4')
```

Out[116]:
```
Name      Shenko
Age           27
Height       5.0
Name: 4, dtype: object
```

20) Insert

When we want to add a new column to a dataframe, it is added at the end by default. However, pandas offers the option to add the new column in any position using insert function.

In [ ]:

In [ ]:

In [ ]: