In [2]:

```
import numpy as np
import pandas as pd
```

In [3]:

```
dict1={'Names':['Ramesh','Suresh',np.nan,'Mahesh'],'Age':[31,32,33,np.nan],'Cit
```

In [8]:

```
data1=pd.DataFrame(dict1)
data1
```

Out[8]:

	Names	Age	City
0	Ramesh	31.0	NaN
1	Suresh	32.0	Hyd
2	NaN	33.0	Mumbai
3	Mahesh	NaN	Chennai

In [7]:

```
data1.isnull()
```

Out[7]:

	Names	Age	City
0	False	False	True
1	False	False	False
2	True	False	False
3	False	True	False

In [9]:

```
data1.isnull().sum()
# it says that every column has a missing value
```

Out[9]:

```
Names     1
Age       1
City      1
dtype: int64
```

In [10]:

```
data1.isnull().sum()/len(data1)
```

Out[10]:

```
Names     0.25
Age       0.25
City      0.25
dtype: float64
```

In [11]:

```
data1.isnull().sum()*100/len(data1)
```

Out[11]:

```
Names     25.0
Age       25.0
City      25.0
dtype: float64
```

In [12]:
```
dict2={'Names':['Ramesh','Suresh',None,'Mahesh'],'Age':[31,32,33,None],'City':[
data2=pd.DataFrame(dict2)
data2
```

Out[12]:

	Names	Age	City
0	Ramesh	31.0	None
1	Suresh	32.0	Hyd
2	None	33.0	Mumbai
3	Mahesh	NaN	Chennai

In [13]:
```
data2.isnull()
```

Out[13]:

	Names	Age	City
0	False	False	True
1	False	False	False
2	True	False	False
3	False	True	False

In [14]:
```
data2.isnull().sum()
```

Out[14]:
```
Names     1
Age       1
City      1
dtype: int64
```

In [15]:
```
data2.isnull().sum()/len(data2)
```

Out[15]:
```
Names     0.25
Age       0.25
City      0.25
dtype: float64
```

In [16]:
```
data2.isnull().sum()*100/len(data2)
```

Out[16]:
```
Names     25.0
Age       25.0
City      25.0
dtype: float64
```

In [17]:
```
dict3={'Names':['Ramesh','Suresh','Null','Mahesh'],'Age':[31,32,33,'Null'],'Cit
data3=pd.DataFrame(dict3)
data3
```

Out[17]:

	Names	Age	City
0	Ramesh	31	Null
1	Suresh	32	Hyd
2	Null	33	Mumbai
3	Mahesh	Null	Chennai

In [18]:
```
data3.isnull()
```

Out[18]:

	Names	Age	City
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False

Method-1

- fill the missing values with random number
- DataFrame name = data1
- method name:fillna

In [19]:
```
data1.fillna(40)
```

Out[19]:

	Names	Age	City
0	Ramesh	31.0	40
1	Suresh	32.0	Hyd
2	40	33.0	Mumbai
3	Mahesh	40.0	Chennai

Method-2

- Fill the missing values with random number on specific column

In [20]:
```
data1['Names'].fillna('Sathish',inplace=True)
data1
```

Out[20]:

	Names	Age	City
0	Ramesh	31.0	NaN
1	Suresh	32.0	Hyd
2	Sathish	33.0	Mumbai
3	Mahesh	NaN	Chennai

In [22]:
```
# Create the data again
dict1={'Names':['Ramesh','Suresh',np.nan,'Mahesh'],'Age':[31,32,33,np.nan],'Cit
data1=pd.DataFrame(dict1)
data1
```

Out[22]:

	Names	Age	City
0	Ramesh	31.0	NaN
1	Suresh	32.0	Hyd
2	NaN	33.0	Mumbai
3	Mahesh	NaN	Chennai

Method-3

- bfill
- ffill
- pad
- backfill

In [23]:
```
data1.fillna(method='backfill')
# Names index 2 is missed value
# it willbe replaced by index 3 vlue
# Age index 3 is misssed  value
#  we dont have index 4 , so the value n NaN
# city index 0 has missed value
#  it replaces with index 1 value
```

Out[23]:

	Names	Age	City
0	Ramesh	31.0	Hyd
1	Suresh	32.0	Hyd
2	Mahesh	33.0	Mumbai
3	Mahesh	NaN	Chennai

In [24]:
```
data1.fillna(method='bfill')
```

Out[24]:

	Names	Age	City
0	Ramesh	31.0	Hyd
1	Suresh	32.0	Hyd
2	Mahesh	33.0	Mumbai
3	Mahesh	NaN	Chennai

In [25]:
```
data1.fillna(method='ffill')
```

Out[25]:

	Names	Age	City
0	Ramesh	31.0	NaN
1	Suresh	32.0	Hyd
2	Suresh	33.0	Mumbai
3	Mahesh	33.0	Chennai

In [26]:
```
data1.fillna(method='pad')
```

Out[26]:

	Names	Age	City
0	Ramesh	31.0	NaN
1	Suresh	32.0	Hyd
2	Suresh	33.0	Mumbai
3	Mahesh	33.0	Chennai

- bfill and backfill both are same
- pad and ffil both are same

Method-4

- mean
- median
- mode

In [28]:
```
age_mean=data1['Age'].mean()
age_mean
```

Out[28]:
```
32.0
```

```
In [30]: data1['Age'].fillna(age_mean)

Out[30]: 0    31.0
         1    32.0
         2    33.0
         3    32.0
         Name: Age, dtype: float64
```

```
In [ ]: # instead of providing a random number
        # we are filling with mean of the data
```

```
In [31]: age_median=data1['Age'].median()
         age_median

Out[31]: 32.0
```

```
In [33]: data1['Age'].fillna(age_median)

Out[33]: 0    31.0
         1    32.0
         2    33.0
         3    32.0
         Name: Age, dtype: float64
```

```
In [34]: age_mode=data1['Age'].mode()
         age_mode

Out[34]: 0    31.0
         1    32.0
         2    33.0
         Name: Age, dtype: float64
```

```
In [35]: data1['Age'].fillna(age_mode)

Out[35]: 0    31.0
         1    32.0
         2    33.0
         3     NaN
         Name: Age, dtype: float64
```

```
In [ ]: # Level1: Mean Median Mode
        # Level2: bfill ffill
        # Lveel3:KNN K nearest negihbours
```

Method 5

KNN imputer

- KNN : K nearest neighbour
- in the KNN imputer instead of taking mean of all the values
- will choose neighbours data and will take those mean only

KNN IMPUTER

In [ ]:
```
n_neighbors
if we do not choose by default it will take as = 5
```

In [38]:
```
from sklearn.impute import KNNImputer
knn=KNNImputer(n_neighbors=2)
knn.fit_transform(data1[['Age']])
```

Out[38]:
```
array([[31.],
       [32.],
       [33.],
       [32.]])
```

In [39]:
```
data1
```

Out[39]:

	Names	Age	City
0	Ramesh	31.0	NaN
1	Suresh	32.0	Hyd
2	NaN	33.0	Mumbai
3	Mahesh	NaN	Chennai

Method-6

- Based on other columns
- sometimes all the above columns will not provide good justification
- at that time we need to check other columns dependency also
- most of the time we will pick the column which has highest correlation

In [ ]:

In [ ]:

In [ ]:

In [ ]: