Data 💻 Handling ▶

```
# Mounting Google Drive for Colab
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# Importing libraries
import pandas as pd
import numpy as np
```

Remove Warning

```
import warnings
warnings.filterwarnings('ignore')
```

Working with missisng Data in Pandas ⟶

→ Missing Data can occur when no information is provided for one or more items or for a whole unit.

→ Missing Data is a very big problem in a real-life scenarios.

→ Missing Data can also refer to as NA(Not Available) values in pandas. In DataFrame sometimes many datasets simply arrive with missing data, either because it exists and was not collected or it never existed.

In Pandas missing data is represented by two value :

→ **None :** None is a Python singleton object that is often used for missing data in Python code.

→ **NaN :** NaN (an acronym for Not a Number), is a special floating-point value recognized by all systems that use the standard IEEE floating-point representation

Pandas treat None and NaN as essentially interchangeable for indicating missing or null values. To facilitate this convention, there are several useful functions for detecting, removing, and replacing null values in Pandas DataFrame :

- isnull()
- notnull()
- dropna()
- fillna()
- replace()
- interpolate()

To better understanding of the concept of Data Handling we'll work on Bengluru House Data.

```
df1 = pd.read_csv('/content/drive/MyDrive/Bengaluru_House_Data.csv')
df1.head()
```

	area_type	availability	location	size	society	total_sqft	bath	balcony	price
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056	2.0	1.0	39.07
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600	5.0	3.0	120.00
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	NaN	1440	2.0	3.0	62.00

Checking for missing values using isnull( ) or isna( ) :

In order to check null values in Pandas DataFrame, we use isnull() or isna() function this function return dataframe of Boolean values which are True for NaN values.

```
df1.isnull().any()        #Boolean output : True/False
```

```
area_type       False
availability    False
location         True
size             True
society          True
total_sqft      False
bath             True
balcony          True
price           False
dtype: bool
```

Numbers of Missing Values (for each column) :

```
df1.isnull().sum()
```

```
area_type          0
availability       0
location           1
size              16
society         5502
total_sqft         0
bath              73
balcony          609
price              0
dtype: int64
```

```
# Total number of missing value:
df1.isna().sum().sum()
```

```
6201
```

```
# Combination of .any() and .sum()
df1.isnull().apply(pd.value_counts)
```

	area_type	availability	location	size	society	total_sqft	bath	balcony	price
False	13320.0	13320.0	13319	13304	7818	13320.0	13247	12711	13320.0
True	NaN	NaN	1	16	5502	NaN	73	609	NaN

```
df1.isnull().apply(pd.value_counts).T
```

	False	True
area_type	13320.0	NaN
availability	13320.0	NaN
location	13319.0	1.0
size	13304.0	16.0
society	7818.0	5502.0
total_sqft	13320.0	NaN
bath	13247.0	73.0
balcony	12711.0	609.0
price	13320.0	NaN

Percentage of Missing value :

```
# Percentage of missing

def per(dataframe):
  a = dataframe.isna().sum()
  perc = (a / (len(dataframe))) *100
  perc = pd.DataFrame(perc,columns = ["%age of missing data"]) #Making DataFrame for better experience
  return perc
per(df1)
```

	%age of missing data
area_type	0.000000
availability	0.000000
location	0.007508
size	0.120120
society	41.306306
total_sqft	0.000000
bath	0.548048

Shape of Dataset :

```
# Shape of data : it will return (rows, columns)
df1.shape
```

```
(13320, 9)
```

```
# Row at the index 0
df1.shape[0]
```

```
13320
```

```
# Column at the index 1
df1.shape[1]
```

```
9
```

Rows ⟹ 11320

Columns ⟹ 9

Filling null values by using .fillna() function :

```
# Filling null values (missing values) with zero
df2 = df1.fillna(0)
# or
df2 = df1.fillna(value = 0)
df2
```

	area_type	availability	location	size	society	total_sqft	bath	balcony	price
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056	2.0	1.0	39.07
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600	5.0	3.0	120.00
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	0	1440	2.0	3.0	62.00
3	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	Soiewre	1521	3.0	1.0	95.00
4	Super built-up Area	Ready To Move	Kothanur	2 BHK	0	1200	2.0	1.0	51.00
...	...	...	...	...	...	...	...	...	...
13315	Built-up Area	Ready To Move	Whitefield	5 Bedroom	ArsiaEx	3453	4.0	0.0	231.00

```
# Checking, does df2 contains null values or not
# df2.isna().sum()
df2.isnull().sum().sum()
```

```
0
```

```
# Dataset df2 contains 0 null values
# Checking for df1
```

```
df1.isna().sum().sum()
     6201
```

```
# But df1 contains 6201 null values.
```

```
# Filling null values with the previous value
df3 = df1.fillna(method = 'pad')
df3
```

	area_type	availability	location	size	society	total_sqft	bath	balcony	price
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056	2.0	1.0	39.07
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600	5.0	3.0	120.00
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	Theanmp	1440	2.0	3.0	62.00
3	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	Soiewre	1521	3.0	1.0	95.00
4	Super built-up Area	Ready To Move	Kothanur	2 BHK	Soiewre	1200	2.0	1.0	51.00
...	...	...	...	...	...	...	...	...	...
13315	Built-up Area	Ready To Move	Whitefield	5 Bedroom	ArsiaEx	3453	4.0	0.0	231.00

```
# Filling null values with coming (next rows) value
df4 = df1.fillna(method = 'bfill')
df4
```

	area_type	availability	location	size	society	total_sqft	bath	balcony	price
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056	2.0	1.0	39.07
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600	5.0	3.0	120.00
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	Soiewre	1440	2.0	3.0	62.00
3	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	Soiewre	1521	3.0	1.0	95.00
4	Super built-up Area	Ready To Move	Kothanur	2 BHK	DuenaTa	1200	2.0	1.0	51.00
...	...	...	...	...	...	...	...	...	...
13315	Built-up Area	Ready To Move	Whitefield	5 Bedroom	ArsiaEx	3453	4.0	0.0	231.00

dropna() function ⟶

Syntax : DataFrame.dropna(self, axis=0, how='any', thresh=None, subset=None, inplace=False)

Parameters :

axis → {0 or 'index', 1 or 'columns'}, default 0 Determine if rows or columns which contain missing values are removed.

0, or 'index' : Drop rows which contain missing values.

1, or 'columns' : Drop columns which contain missing value.

how → {'any', 'all'}, default

any : Determine if row or column is removed from DataFrame, when we have at least one NA or all NA.

`'any'` : If any NA values are present, drop that row or column.

`'all'` : If all values are NA, drop that row or column.

---

thres → hint, optional : Require that many non-NA values. Cannot be combined with how.

subset → column label or sequence of labels, optional : Labels along other axis to consider, e.g. if you are dropping rows these would be a list of columns to include.

inplace → bool, default False : Whether to modify the DataFrame rather than creating a new one.

Returns : DataFrame or None DataFrame with NA entries dropped from it or None if *`inplace=True`*.

dfropna() ⟶

→ The dropna() method removes the rows that contains NULL values.

→ The dropna() method returns a new DataFrame object unless the inplace parameter is set to True , in that case the dropna() method does the removing in the original DataFrame instead.

```
# it can be run everytime we want to run this cell.
# But if we used inplace = True, it will show error because it will be deleted from original dataFrame.
df5 = df1.dropna()
df5.isna().sum()
```

```
area_type       0
availability    0
location        0
size            0
society         0
total_sqft      0
bath            0
balcony         0
price           0
dtype: int64
```

dropna(how) ⟶

how = 'all' → It will remove rows contains all null values.

how = 'any' → It will remove all rows which have any (at least a single) null value.

```
# how = 'all'
df6 = df1.dropna(how='all')
df6.isna().sum()
```

```
area_type          0
availability       0
location           1
size              16
society         5502
total_sqft         0
bath              73
balcony          609
price              0
dtype: int64
```

```
df7 = df1.dropna(how='any')
df7.isna().sum()
```

```
area_type       0
availability    0
location        0
size            0
society         0
total_sqft      0
bath            0
balcony         0
price           0
dtype: int64
```

.replace() ⟶

→ We can replace and fill also

```
df8 = df1.replace(to_replace = np.nan, value = 1234)
# Null values will be replaced by 1234. (any rows or columns)
df8
```

	area_type	availability	location	size	society	total_sqft	bath	balcony	price
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056	2.0	1.0	39.07
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600	5.0	3.0	120.00
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	1234	1440	2.0	3.0	62.00
3	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	Soiewre	1521	3.0	1.0	95.00
4	Super built-up Area	Ready To Move	Kothanur	2 BHK	1234	1200	2.0	1.0	51.00
...	...	...	...	...	...	...	...	...	...
13315	Built-up Area	Ready To Move	Whitefield	5 Bedroom	ArsiaEx	3453	4.0	0.0	231.00

```
df = pd.read_csv('/content/drive/MyDrive/Weather.csv')
df.head()
```

	STA	Date	Precip	WindGustSpd	MaxTemp	MinTemp	MeanTemp	Snowfall	PoorWeather	YR	...
0	10001	7/1/1942	1.016	NaN	25.555556	22.222222	23.888889	0.0	NaN	42	...
1	10001	7/2/1942	0	NaN	28.888889	21.666667	25.555556	0.0	NaN	42	...
2	10001	7/3/1942	2.54	NaN	26.111111	22.222222	24.444444	0.0	NaN	42	...
3	10001	7/4/1942	2.54	NaN	26.666667	22.222222	24.444444	0.0	NaN	42	...
4	10001	7/5/1942	0	NaN	26.666667	21.666667	24.444444	0.0	NaN	42	...

5 rows × 30 columns

```
## checking for missing value using isnull() or isna()
df.isnull().any()        # Boolean Output => True or False
```

```
STA            False
Date           False
Precip         False
WindGustSpd     True
MaxTemp        False
MinTemp        False
MeanTemp       False
Snowfall        True
PoorWeather     True
YR             False
MO             False
DA             False
PRCP            True
DR              True
SPD             True
MAX             True
MIN             True
MEA             True
SNF             True
SND             True
FT              True
FB              True
FTI             True
PGT             True
TSHDSBRSGF      True
SD3             True
RHX             True
RHN             True
RVG             True
WTE             True
dtype: bool
```

Number of missing value :

```
# Number of missing values
df.isnull().sum()
```

```
STA                 0
Date                0
Precip              0
WindGustSpd    118508
MaxTemp             0
MinTemp             0
MeanTemp            0
Snowfall         1163
PoorWeather     84803
YR                  0
MO                  0
DA                  0
PRCP             1932
DR             118507
SPD            118508
MAX               474
MIN               468
MEA               498
SNF              1163
SND            113477
FT             119040
FB             119040
FTI            119040
PGT            118515
TSHDSBRSGF      84803
SD3            119040
RHX            119040
RHN            119040
RVG            119040
WTE            119040
dtype: int64
```

Total Number of Missing Values

```
df.isnull().sum().sum()
```

```
1715139
```

```
# Combination of .any() and .sum()
df.isnull().apply(pd.value_counts).T
```

	False	True
STA	119040.0	NaN
Date	119040.0	NaN
Precip	119040.0	NaN
WindGustSpd	532.0	118508.0
MaxTemp	119040.0	NaN
MinTemp	119040.0	NaN

Percentage of missing values:

```
per(df)
```

	%age of missing data
STA	0.000000
Date	0.000000
Precip	0.000000
WindGustSpd	99.553091
MaxTemp	0.000000
MinTemp	0.000000
MeanTemp	0.000000
Snowfall	0.976983
PoorWeather	71.239079
YR	0.000000
MO	0.000000
DA	0.000000
PRCP	1.622984
DR	99.552251
SPD	99.553091
MAX	0.398185
MIN	0.393145
MEA	0.418347
SNF	0.976983
SND	95.326781
FT	100.000000
FB	100.000000
FTI	100.000000
PGT	99.558972
TSHDSBRSGF	71.239079
SD3	100.000000
RHX	100.000000
RHN	100.000000
RVG	100.000000
WTE	100.000000

```
df.shape
```

```
(119040, 30)
```

Why Do We Need To Care About Handling Missing Value?

It is important to handle the missing values appropriately.

- Many machine learning algorithms fail if the dataset contains missing values. However, algorithms like K-nearest and Naive Bayes support data with missing values.
- You may end up building a biased machine learning model which will lead to incorrect results if the missing values are not handled properly.
- Missing data can lead to a lack of precision in the statistical analysis.

Dropping / Deleting of a single column :

Example :

→ **WindGustSpd** has missing 99.553091 missing data, if python will automate or fill these missing data, it will be statistically wrong.

So we'll delete the **WindGustSpd** column.

Dropping from root level use **inplace = True**

```
#Dropping / Deleting of a single column
# df.drop('WindGustSpd',axis = 1, inplace = True)
```

```
df.head()
```

	STA	Date	Precip	WindGustSpd	MaxTemp	MinTemp	MeanTemp	Snowfall	PoorWeather	YR	...
0	10001	7/1/1942	1.016	NaN	25.555556	22.222222	23.888889	0.0	NaN	42	...
1	10001	7/2/1942	0	NaN	28.888889	21.666667	25.555556	0.0	NaN	42	...
2	10001	7/3/1942	2.54	NaN	26.111111	22.222222	24.444444	0.0	NaN	42	...
3	10001	7/4/1942	2.54	NaN	26.666667	22.222222	24.444444	0.0	NaN	42	...
4	10001	7/5/1942	0	NaN	26.666667	21.666667	24.444444	0.0	NaN	42	...

5 rows × 30 columns

```
df.shape
```

```
(119040, 30)
```

```
df.columns
```

```
Index(['STA', 'Date', 'Precip', 'WindGustSpd', 'MaxTemp', 'MinTemp',
       'MeanTemp', 'Snowfall', 'PoorWeather', 'YR', 'MO', 'DA', 'PRCP', 'DR',
       'SPD', 'MAX', 'MIN', 'MEA', 'SNF', 'SND', 'FT', 'FB', 'FTI', 'PGT',
       'TSHDSBRSGF', 'SD3', 'RHX', 'RHN', 'RVG', 'WTE'],
      dtype='object')
```

So column 'WingGustSpd' is deleted.

Dropping / Deleting Multiple Columns ⟶

Dropping columns, if a column contains maximum null values like above 30 % - 40 %.

```
# df.drop(['PoorWeather','DR','SPD','SND','FT', 'FB', 'FTI', 'PGT', 'TSHDSBRSGF', 'SD3',
#         'RHX', 'RHN', 'RVG', 'WTE'], axis = 1, inplace = True)
```

```
df.shape
```

```
(119040, 30)
```

→ Only 15 columns left, which have either minimum missing value (lower than 40 %) or no missing values.

```
# Percentage of missing values
per(df)
```

	%age of missing data
STA	0.000000
Date	0.000000
Precip	0.000000
WindGustSpd	99.553091
MaxTemp	0.000000
MinTemp	0.000000
MeanTemp	0.000000
Snowfall	0.976983
PoorWeather	71.239079
YR	0.000000
MO	0.000000
DA	0.000000
PRCP	1.622984
DR	99.552251
SPD	99.553091
MAX	0.398185
MIN	0.393145
MEA	0.418347
SNF	0.976983
SND	95.326781
FT	100.000000
FB	100.000000
FTI	100.000000
PGT	99.558972
TSHDSBRSGF	71.239079
SD3	100.000000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119040 entries, 0 to 119039
Data columns (total 30 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   STA          119040 non-null  int64
 1   Date         119040 non-null  object
 2   Precip       119040 non-null  object
 3   WindGustSpd  532 non-null     float64
 4   MaxTemp      119040 non-null  float64
 5   MinTemp      119040 non-null  float64
 6   MeanTemp     119040 non-null  float64
 7   Snowfall     117877 non-null  object
 8   PoorWeather  34237 non-null   object
 9   YR           119040 non-null  int64
 10  MO           119040 non-null  int64
 11  DA           119040 non-null  int64
 12  PRCP         117108 non-null  object
 13  DR           533 non-null     float64
 14  SPD          532 non-null     float64
 15  MAX          118566 non-null  float64
 16  MIN          118572 non-null  float64
 17  MEA          118542 non-null  float64
 18  SNF          117877 non-null  object
 19  SND          5563 non-null    float64
 20  FT           0 non-null       float64
 21  FB           0 non-null       float64
 22  FTI          0 non-null       float64
 23  PGT          525 non-null     float64
 24  TSHDSBRSGF   34237 non-null   object
 25  SD3          0 non-null       float64
 26  RHX          0 non-null       float64
 27  RHN          0 non-null       float64
 28  RVG          0 non-null       float64
 29  WTE          0 non-null       float64
dtypes: float64(19), int64(4), object(7)
memory usage: 27.2+ MB
```

Fill these missing values with **median** using **.fillna()** method:

```
# Median of MAX column
df['MAX'].median()
```

```
85.0
```

```
# Filling column name 'MAX' data with its median
df['MAX'].fillna(df['MAX'].median(), inplace = True)

# Filling column name 'MIN' data with its median
df['MIN'].fillna(df['MIN'].median(), inplace = True)

# Filling column name 'MEA' data with its median
df['MEA'].fillna(df['MEA'].median(), inplace = True)

# These all have datatype either int or float, so these get easily filled with median.
```

These all have datatype either int or float, so these get easily filled with median.

```
#Check which column still have missing values
df.isna().sum()
```

```
STA                 0
Date                0
Precip              0
WindGustSpd    118508
MaxTemp             0
MinTemp             0
MeanTemp            0
Snowfall         1163
PoorWeather     84803
YR                  0
MO                  0
DA                  0
PRCP             1932
DR             118507
SPD            118508
MAX                 0
MIN                 0
MEA                 0
SNF              1163
SND            113477
FT             119040
FB             119040
FTI            119040
PGT            118515
TSHDSBRSGF      84803
SD3            119040
RHX            119040
RHN            119040
RVG            119040
WTE            119040
dtype: int64
```

Snowfall, PRCP and **SNF** still contain missing values.

All of the above contain 'object' datatype.

1. Check their value counts.
2. Find Object datatype and replace it using .replace() with NaN value using np.nan.
3. Change its datatype to float using .astype()
4. Now fill the missing values with median.

```
df['Snowfall'].value_counts()
```

```
0.0        86090
0          29600
5.08         527
7.62         319
2.54         317
10.16        195
12.7          90
20.32         83
17.78         78
15.24         70
22.86         69
25.4          68
```

```
#VALUE!       44
27.94         40
30.48         31
45.72         25
50.8          24
48.26         22
2.54          22
35.56         20
33.02         15
60.96         13
7.62          11
38.1          11
66.04         11
53.34         10
43.18         10
10.16         10
63.5           7
5.08           7
55.88          6
40.64          6
76.2           5
58.42          5
15.24          4
81.28          4
78.74          2
12.7           2
83.82          1
68.58          1
86.36          1
73.66          1
Name: Snowfall, dtype: int64
```

Here **#VALUE!** has object datatype and contains 44 missing values.

Replace these values with NaN :

```
#Here #VALUE! has object datatype and contains 44 missing values.
# Replacing these values with NaN

df['Snowfall'] = df['Snowfall'].replace('#VALUE!', np.nan)
```

Changing its datatype objectto float :

```
# Changing its datatype objectto float
df['Snowfall'] = df['Snowfall'].astype('float')
```

Filling the NaN (missing) values with median :

```
# Filling the NaN (missing) values with median :

df['Snowfall'].fillna(df['Snowfall'].median(), inplace = True)
```

Let's do same steps for **PRCP** and **SNF**

```
df['PRCP'].value_counts()
```

```
0       62335
T       16753
0.01     3389
0.02     2909
0.03     2015
         ...
4.87        1
4.2         1
4.98        1
4.88        1
6.34        1
Name: PRCP, Length: 540, dtype: int64
```

```
# For PRCP column
df['PRCP'] = df['PRCP'].replace('T', np.nan)
df['PRCP'] = df['PRCP'].astype('float')
df['PRCP'].fillna(df['PRCP'].median(), inplace = True)
```

```
df['SNF'].value_counts()
```

```
0.0    86090
0      29600
```

```
0.2      527
0.3      319
0.1      317
0.4      195
0.5       90
0.8       83
0.7       78
0.6       70
0.9       69
1         68
T         44
1.1       40
1.2       31
1.8       25
2         24
1.9       22
0.1       22
1.4       20
1.3       15
2.4       13
0.3       11
1.5       11
2.6       11
2.1       10
1.7       10
0.4       10
2.5        7
0.2        7
2.2        6
1.6        6
3          5
2.3        5
0.6        4
3.2        4
3.1        2
0.5        2
3.3        1
2.7        1
3.4        1
2.9        1
Name: SNF, dtype: int64
```

```
# For SNF column
df['SNF'] = df['SNF'].replace('T', np.nan)
df['SNF'] = df['SNF'].astype('float')
df['SNF'].fillna(df['SNF'].median(), inplace = True)
```

```
# Check missing value
df.isna().sum()
```

```
STA                 0
Date                0
Precip              0
WindGustSpd    118508
MaxTemp             0
MinTemp             0
MeanTemp            0
Snowfall            0
PoorWeather     84803
YR                  0
MO                  0
DA                  0
PRCP                0
DR             118507
SPD            118508
MAX                 0
MIN                 0
MEA                 0
SNF                 0
SND            113477
FT             119040
FB             119040
FTI            119040
PGT            118515
TSHDSBRSGF      84803
SD3            119040
RHX            119040
RHN            119040
RVG            119040
WTE            119040
dtype: int64
```

No missing values present in the dataset.

```
per(df)
```

	%age of missing data
STA	0.000000
Date	0.000000
Precip	0.000000
WindGustSpd	99.553091
MaxTemp	0.000000
MinTemp	0.000000
MeanTemp	0.000000
Snowfall	0.000000
PoorWeather	71.239079
YR	0.000000
MO	0.000000
DA	0.000000
PRCP	0.000000
DR	99.552251
SPD	99.553091
MAX	0.000000
MIN	0.000000
MEA	0.000000
SNF	0.000000
SND	95.326781
FT	100.000000
FB	100.000000
FTI	100.000000
PGT	99.558972
TSHDSBRSGF	71.239079
SD3	100.000000
RHX	100.000000
RHN	100.000000
RVG	100.000000
WTE	100.000000

Percentage of missing values in data is 0 %.

Now it is a clean dataset.