```
import pandas as pd
#from google.colab import drive
#drive.mount('/content/drive')

df = pd.read_csv("https://raw.githubusercontent.com/dataoptimal/posts/master/data%20cle
df
#https://raw.githubusercontent.com/dataoptimal/posts/master/data%20cleaning%20with%20py
```

	PID	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
0	100001000.0	104.0	PUTNAM	Y	3	1	1000
1	100002000.0	197.0	LEXINGTON	N	3	1.5	--
2	100003000.0	NaN	LEXINGTON	N	NaN	1	850
3	100004000.0	201.0	BERKELEY	12	1	NaN	700
4	NaN	203.0	BERKELEY	Y	3	2	1600
5	100006000.0	207.0	BERKELEY	Y	NaN	1	800
6	100007000.0	NaN	WASHINGTON	NaN	2	HURLEY	950
7	100008000.0	213.0	TREMONT	Y	1	1	NaN
8	100009000.0	215.0	TREMONT	Y	na	2	1800

```
# Standard Missing Values
df.dtypes
```

```
PID             float64
ST_NUM          float64
ST_NAME          object
```

```
df['ST_NUM'].isnull()
```

```
0    False
1    False
2     True
3    False
4    False
5    False
6     True
7    False
8    False
Name: ST_NUM, dtype: bool
```

```
df['SQ_FT'].isnull()
```

```
0    False
1    False
2    False
3    False
4    False
5    False
6    False
7     True
8    False
Name: SQ_FT, dtype: bool
```

```
df['NUM_BEDROOMS'].isnull()
```

```
0    False
1    False
2     True
3    False
```

```
# Making a list of missing value types
missing_values = ["n/a", "na", "--"]
df = pd.read_csv("https://raw.githubusercontent.com/dataoptimal/posts/master/data%20cle
                 na_values = missing_values)
```

Double-click (or enter) to edit

```
df
```

	PID	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
0	100001000.0	104.0	PUTNAM	Y	3.0	1	1000.0
1	100002000.0	197.0	LEXINGTON	N	3.0	1.5	NaN
2	100003000.0	NaN	LEXINGTON	N	NaN	1	850.0
3	100004000.0	201.0	BERKELEY	12	1.0	NaN	700.0
4	NaN	203.0	BERKELEY	Y	3.0	2	1600.0
5	100006000.0	207.0	BERKELEY	Y	NaN	1	800.0
6	100007000.0	NaN	WASHINGTON	NaN	2.0	HURLEY	950.0
7	100008000.0	213.0	TREMONT	Y	1.0	1	NaN
8	100009000.0	215.0	TREMONT	Y	NaN	2	1800.0

```
df['NUM_BEDROOMS'].isnull()
```

```
0    False
1    False
2     True
3    False
4    False
5     True
6    False
7    False
```

```
import numpy as np
# Detecting numbers
cnt=0
for row in df['OWN_OCCUPIED']:
    try:
        int(row)
        df.loc[cnt, 'OWN_OCCUPIED']=np.nan
    except ValueError:
        pass
    cnt+=1
```

```
df
```

	PID	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
0	100001000.0	104.0	PUTNAM	Y	3.0	1	1000.0

```
import numpy as np
df['OWN_OCCUPIED'] = df['OWN_OCCUPIED'].replace(r'[0-9]+', np.nan, regex=True)

df['NUM_BATH'] = df['NUM_BATH'].replace(r'[A-Za-z]+', np.nan, regex=True)
```

Double-click (or enter) to edit

6	100007000.0	NaN	WASHINGTON	NaN	2.0	HURLEY	950.0
8	100009000.0	215.0	TREMONT	Y	NaN	2	1800.0

```
df
```

	PID	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
0	100001000.0	104.0	PUTNAM	Y	3.0	1	1000.0
1	100002000.0	197.0	LEXINGTON	N	3.0	1.5	NaN
2	100003000.0	NaN	LEXINGTON	N	NaN	1	850.0
3	100004000.0	201.0	BERKELEY	NaN	1.0	NaN	700.0
4	NaN	203.0	BERKELEY	Y	3.0	2	1600.0
5	100006000.0	207.0	BERKELEY	Y	NaN	1	800.0
6	100007000.0	NaN	WASHINGTON	NaN	2.0	NaN	950.0
7	100008000.0	213.0	TREMONT	Y	1.0	1	NaN
8	100009000.0	215.0	TREMONT	Y	NaN	2	1800.0

```
# Total missing values for each feature NaN,np.nan
df.isnull().sum()
```

```
PID             1
ST_NUM          2
ST_NAME         0
OWN_OCCUPIED    2
NUM_BEDROOMS    3
NUM_BATH        2
SQ_FT           2
dtype: int64
```

```
# Any missing values?
#df.isnull().values.any()
```

```
True
```

```
# Total number of missing values
df.isnull().sum().sum()
```

```
12
```

```
# filling missing value using fillna()
df.fillna(0)
```

	PID	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
0	100001000.0	104.0	PUTNAM	Y	3.0	1	1000.0

```
# filling  null value using fillna() function
df.fillna(method ='bfill')  ffill
```

	PID	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
0	100001000.0	104.0	PUTNAM	Y	3.0	1	1000.0
1	100002000.0	197.0	LEXINGTON	N	3.0	1.5	850.0
2	100003000.0	201.0	LEXINGTON	N	1.0	1	850.0
3	100004000.0	201.0	BERKELEY	Y	1.0	2	700.0
4	100006000.0	203.0	BERKELEY	Y	3.0	2	1600.0
5	100006000.0	207.0	BERKELEY	Y	2.0	1	800.0
6	100007000.0	213.0	WASHINGTON	Y	2.0	1	950.0
7	100008000.0	213.0	TREMONT	Y	1.0	1	1800.0
8	100009000.0	215.0	TREMONT	Y	NaN	2	1800.0

```
# will replace  Nan value in dataframe with value -99
df.replace(to_replace = np.nan, value = -99)

fillna(-99)
```

	PID	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
0	100001000.0	104.0	PUTNAM	Y	3.0	1	1000.0
1	100002000.0	197.0	LEXINGTON	N	3.0	1.5	-99.0
2	100003000.0	-99.0	LEXINGTON	N	-99.0	1	850.0
3	100004000.0	201.0	BERKELEY	-99	1.0	-99	700.0
4	-99.0	203.0	BERKELEY	Y	3.0	2	1600.0

```
# using dropna() function
df.dropna()
```

	PID	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
0	100001000.0	104.0	PUTNAM	Y	3.0	1	1000.0

```
# using dropna() function
df.dropna(how = 'all')
```

	PID	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
0	100001000.0	104.0	PUTNAM	Y	3.0	1	1000.0
1	100002000.0	197.0	LEXINGTON	N	3.0	1.5	NaN
2	100003000.0	NaN	LEXINGTON	N	NaN	1	850.0
3	100004000.0	201.0	BERKELEY	NaN	1.0	NaN	700.0
4	NaN	203.0	BERKELEY	Y	3.0	2	1600.0
5	100006000.0	207.0	BERKELEY	Y	NaN	1	800.0
6	100007000.0	NaN	WASHINGTON	NaN	2.0	NaN	950.0
7	100008000.0	213.0	TREMONT	Y	1.0	1	NaN
8	100009000.0	215.0	TREMONT	Y	NaN	2	1800.0

```
# using dropna() function
df.dropna(axis = 1)
```

	ST_NAME
0	PUTNAM
1	LEXINGTON
2	LEXINGTON
3	BERKELEY
4	BERKELEY
5	BERKELEY
6	WASHINGTON
7	TREMONT
8	TREMONT

```
# making new data frame with dropped NA values
new_data = df.dropna(axis = 0, how ='any')


# importing pandas as pd
import pandas as pd

# importing numpy as np
import numpy as np

# dictionary of lists
dict = {'First Score':[100, 90, np.nan, 95],
        'Second Score': [30, 45, 56, np.nan],
        'Third Score':[np.nan, 40, 80, 98]}
```

```
            third score ':[np.nan, 40, 80, 98]}

# creating a dataframe from list
df = pd.DataFrame(dict)

# using isnull() function
df.isnull()
```

	First Score	Second Score	Third Score
0	False	False	True
1	False	False	False
2	True	False	False
3	False	True	False

```
import pandas as pd
# load the dataset
dataset = pd.read_csv('https://gist.githubusercontent.com/ktisha/c21e73a1bd1700294ef790
# print the first 20 rows of data
#https://raw.githubusercontent.com/rrichajalota/Pima-Indians-Diabetes-kaggle/master/dia
dataset.head(20)
```

	0	1	2	3	4	5	6	7	8
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1
10	4	110	92	0	0	37.6	0.191	30	0
11	10	168	74	0	0	38.0	0.537	34	1
12	10	139	80	0	0	27.1	1.441	57	0

```
# count the number of missing values for each column
num_missing = (dataset[[1,2,3,4,5]] == 0).sum()
# report the results
print(num_missing)
```

```
1      5
2     35
3    227
4    374
5     11
dtype: int64
```

```
import numpy as np
```

```
# replace '0' values with 'nan'
dataset[[1,2,3,4,5]] = dataset[[1,2,3,4,5]].replace(0, np.nan)
# count the number of nan values in each column
print(dataset.isnull().sum())
```

```
0      0
1      5
2     35
3    227
4    374
5     11
6      0
7      0
8      0
dtype: int64
```

```
dataset.head()
```

	0	1	2	3	4	5	6	7	8
0	6	148.0	72.0	35.0	NaN	33.6	0.627	50	1
1	1	85.0	66.0	29.0	NaN	26.6	0.351	31	0
2	8	183.0	64.0	NaN	NaN	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1

```
# drop rows with missing values
dataset.dropna(inplace=True)
```

```
# fill missing values with mean column values
dataset.fillna(dataset.mean(), inplace=True)
```

```
dataset = pd.DataFrame({'Column1': transformed_values[:, 0], 'Column2': transformed_val
```

```
dataset.head()
```

- missing_values : The missing_values placeholder which has to be imputed. By default is NaN
- stategy : The data which will replace the NaN values from the dataset. The strategy argument can take the values – ‘mean'(default), ‘median’, ‘most_frequent’ and ‘constant’.
- fill_value : The constant value to be given to the NaN data using the constant strategy.

```
dataset.head(10)
```

	0	1	2	3	4	5	6	7	8
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26	1
8	2	197.0	70.0	45.0	543.0	30.5	0.158	53	1
13	1	189.0	60.0	23.0	846.0	30.1	0.398	59	1
14	5	166.0	72.0	19.0	175.0	25.8	0.587	51	1
16	0	118.0	84.0	47.0	230.0	45.8	0.551	31	1
18	1	103.0	30.0	38.0	83.0	43.3	0.183	33	0
19	1	115.0	70.0	30.0	96.0	34.6	0.529	32	1
20	3	126.0	88.0	41.0	235.0	39.3	0.704	27	0

```
from sklearn.impute import SimpleImputer

dataset = pd.read_csv('https://gist.githubusercontent.com/ktisha/c21e73a1bd1700294ef790
# mark zero values as missing or NaN
dataset[[1,2,3,4,5]] = dataset[[1,2,3,4,5]].replace(0, np.nan)
# retrieve the numpy array
values = dataset.values
# define the imputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
# transform the dataset
transformed_values = imputer.fit_transform(values)
```