

▼ Data Cleaning is the process of preventing and correcting the errors such as duplicates, missing values and incomplete data by importing libraries pandas and numpy

```
import pandas as pd
import numpy as np
#Coding for importing csv files in Google colab
from google.colab import files
import io
uploaded = files.upload()
df = pd.read_csv(io.BytesIO(uploaded['property_data.csv']))
# Read csv file property_data.csv into a pandas dataframe

# Take a look at the first few rows
print(df)
```



Choose Files

No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving property_data.csv to property_data.csv

	ST_NUM	ST_NAME	NUM_ROOMS	OWN_OCCUPIED
0	104.0	LEXTON	2	Y
1	197.0	BERKELEY	3	N
2	NaN	WASHINTON	3	Y
3	201.0	TREMONT	NaN	N
4	203.0	TREMONT	na	12
5	205.0	TREMONT	--	Y
6	NaN	BLAZE	4	NaN
7	213.0	BLAZE	5	N
8	214.0	BLAZE	6	N

▼ Looking at the ST_NUM and NUM_ROOMS columns whether they have any null values

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

```
print(df['NUM_ROOMS'].isnull())  
print(df['NUM_ROOMS'])  
print(df['NUM_ROOMS'].isnull())
```



```
0    104.0
1    197.0
```

▼ To mention possible missing values while opening the file

```
print(df['NUM_ROOMS'])
print(df['NUM_ROOMS'].isnull())
```

```
0    2
1    3
2    3
3    NaN
4    na
5    --
6    4
7    5
8    6
Name: NUM_ROOMS, dtype: object
0    False
1    False
2    False
3     True
4    False
5    False
6    False
7    False
8    False
Name: NUM_ROOMS, dtype: bool

4    False
```

▼ The above 3,4 5th index is not numeric whih needs correction again therefore we need coding part for it. It is corrected with coding below

```
# Detecting numbers and replace it with NaN of numpy
```

```
cnt=0
for row in df['NUM_ROOMS']:
    try:
        int(row)
    except ValueError:
        df.loc[cnt, 'NUM_ROOMS']=np.nan
    cnt+=1
print(df['NUM_ROOMS'])
print(df['NUM_ROOMS'].isnull())
```

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

```
print(df['OWN_OCCUPIED'])
print(df['OWN_OCCUPIED'].isnull())
```

```
0
```

```
0      Y
1      N
2      Y
3      N
4     12
5      Y
6     NaN
7      N
8      N
Name: OWN_OCCUPIED, dtype: object
0     False
1     False
2     False
3     False
```

▼ The above 4 th index is numeric which needs correction again therefore we need coding part for it. It is corrected with coding below

```
# Detecting numbers and replace it with NaN of numpy
cnt=0
for row in df['OWN_OCCUPIED']:
    try:
        int(row)
        df.loc[cnt, 'OWN_OCCUPIED']=np.nan
    except ValueError:
        pass
    cnt+=1
print(df['OWN_OCCUPIED'])
print(df['OWN_OCCUPIED'].isnull())
```



```

0      Y
1      N
2      Y
3      N
4     NaN
5      Y
6     NaN
7      N
8      N
Name: OWN_OCCUPIED, dtype: object
0     False
1     False
2     False

```

```

# Total missing values for each feature
print(df.isnull().sum())
print('Total missing values')
print(sum(df.isnull().sum()))

```

```

↳ ST_NUM      2
   ST_NAME     0
   NUM_ROOMS   3
   OWN_OCCUPIED 2
dtype: int64
Total missing values
7

```

▼ The coding below is an alternative to the above one.

```

# Finding TOTAL missing of each mission fields and Total number of missing values
print(df['ST_NUM'].isnull().sum())
print(df['ST_NAME'].isnull().sum())
print(df['NUM_ROOMS'].isnull().sum())
print(df['OWN_OCCUPIED'].isnull().sum())
print(df.isnull().sum().sum())

```

```
↳ 2
0
3
2
7
```

```
# Finding COUNT AND average of missing of each mission fields and Total number of missing values
df.shape
```

```
print(df['ST_NUM'].isnull().sum()/9.0)
print(df['ST_NAME'].isnull().sum()/9.0)
print(df['NUM_ROOMS'].isnull().sum()/9.0)
print(df['OWN_OCCUPIED'].isnull().sum()/9.0)
```

```
↳ 0.2222222222222222
0.0
0.3333333333333333
0.1111111111111111
```

```
# error percentages are greater than 10%
# Replace missing values with a number
df['ST_NUM'].fillna(125, inplace=True)
print('ST_NUM')
print(df['ST_NUM'])
```

```
↳ ST_NUM
0    104.0
1    197.0
2    125.0
3    201.0
4    203.0
5    205.0
6    125.0
7    213.0
8    214.0
Name: ST_NUM, dtype: float64
```

```
# Replace using media
median = df['NUM_ROOMS'].median()
```

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

```
0      2
1      3
2      3
3     3.5
4     3.5
5     3.5
6      4
7      5
8      6
Name: NUM_ROOMS, dtype: object
```

```
print(df)
```

```
0      ST_NUM  ST_NAME  NUM_ROOMS  OWN_OCCUPIED
1      104.0    LEXTON          2             Y
2      197.0   BERKELEY          3             N
3      125.0  WASHINGTON          3             Y
4      201.0   TREMONT         3.5             N
5      203.0   TREMONT         3.5            NaN
6      205.0   TREMONT         3.5             Y
7      125.0    BLAZE          4            NaN
8      213.0    BLAZE          5             N
9      214.0    BLAZE          6             N
```

```
# to find the maximum occurrence of the values in the attribute OWN_OCCUPIED
df[ 'OWN_OCCUPIED' ].value_counts().idxmax()
```

```
'N'
```

```
df
```


	ST_NUM	ST_NAME	NUM_ROOMS	OWN_OCCUPIED
0	104.0	LEXTON	2	Y
1	197.0	BERKELEY	3	N
2	125.0	WASHINTON	3	Y
3	201.0	TREMONT	3.5	N
4	203.0	TREMONT	3.5	NaN
5	205.0	TREMONT	3.5	Y
6	125.0	BLAZE	4	NaN
7	213.0	BLAZE	5	N

```
# finding all value occurrence
df['OWN_OCCUPIED'].value_counts()
```

```
↳ N    4
   Y    3
   Name: OWN_OCCUPIED, dtype: int64
```

```
# Replacing all NaN in ST_NUM with some random value 125
df['ST_NUM'].fillna(125)
```

```
↳ 0    104.0
   1    197.0
   2    125.0
   3    201.0
   4    203.0
   5    205.0
   6    125.0
   7    213.0
   8    214.0
   Name: ST_NUM, dtype: float64
```

```
df
```



	ST_NUM	ST_NAME	NUM_ROOMS	OWN_OCCUPIED
0	104.0	LEXTON	2	Y
1	197.0	BERKELEY	3	N
2	125.0	WASHINTON	3	Y
3	201.0	TREMONT	3.5	N
4	203.0	TREMONT	3.5	NaN
5	205.0	TREMONT	3.5	Y
6	125.0	BLAZE	4	NaN
7	213.0	BLAZE	5	N
8	214.0	BLAZE	6	N

```
# Replacing all NaN in own_occupied with N is justified as its occurrene is more  
df['OWN_OCCUPIED'].fillna('N',inplace=True)
```

```
df
```



we have got a cleaned data set now

2	125.0	WASHINTON	3	Y
3	201.0	TREMONT	3.5	N
4	203.0	TREMONT	3.5	N
5	205.0	TREMONT	3.5	Y
6	125.0	BLAZE	4	N
7	213.0	BLAZE	5	N
8	214.0	BLAZE	6	N