EDA 02 :Handling Missing Values

1. Necessary imports and uplaoding the csv

In [209]:
```
import pandas as pd
import numpy as np
```

In [210]:
```
data=pd.read_csv("D:/FTI/Cohort 2 EDA/Lecture 2/property.csv")
```

2. Know Your Data - Data Description

Data Description

1. PID: Property ID
2. ST_NUM : Street Number
3. ST_NAME: Street Name
4. OWN_OCCUPIED : VACANT or OCCUPIED
5. NUM_BEDROOMS: Number of Bed Rooms
6. NUM_BATH: Number of Bathrooms

7 SQ_FT : Area in Feet

	A	B	C	D	E	F	G
1	PID	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
2	100001000	104	PUTNAM	Y	3	1	1000
3	100002000	197	LEXINGTON	N	3	1.5	--
4	100003000		LEXINGTON	N	n/a	1	850
5	100004000	201	BERKELEY	12	1	NaN	700
6		203	BERKELEY	Y	3	2	1600
7	100006000	207	BERKELEY	Y	NA	1	800
8	100007000	NA	WASHINGTON		2	HURLEY	950
9	100008000	213	TREMONT	Y	1	1	
10	100009000	215	TREMONT	Y	na	2	1800

In [211]:
```
data.shape
```

Out[211]: (9, 7)

In [212]:
```
data
```

Out[212]:

	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

In [213]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9 entries, 0 to 8
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   PID            8 non-null      float64
 1   ST_NUM         7 non-null      float64
 2   ST_NAME        9 non-null      object 
 3   OWN_OCCUPIED   8 non-null      object 
 4   NUM_BEDROOMS   7 non-null      object 
 5   NUM_BATH       8 non-null      object 
 6   SQ_FT          8 non-null      object 
dtypes: float64(2), object(5)
memory usage: 632.0+ bytes
```

In [214]:
```
data.describe()
```

Out[214]:

	PID	ST_NUM
count	8.000000e+00	7.000000
mean	1.000050e+08	191.428571
std	2.927700e+03	39.080503
min	1.000010e+08	104.000000
25%	1.000028e+08	199.000000
50%	1.000050e+08	203.000000
75%	1.000072e+08	210.000000
max	1.000090e+08	215.000000

In [215]:
```
data.dtypes
```

Out[215]:
```
PID             float64
ST_NUM          float64
ST_NAME          object
OWN_OCCUPIED     object
NUM_BEDROOMS     object
NUM_BATH         object
SQ_FT            object
dtype: object
```

3. Checking for NULL Values

In [216]:
```
 data.isnull().sum()
```

Out[216]:
```
PID             1
ST_NUM          2
ST_NAME         0
OWN_OCCUPIED    1
NUM_BEDROOMS    2
NUM_BATH        1
SQ_FT           1
dtype: int64
```

data.isnull.sum() returns the total number of missing values in all columns. Values Recognized as Missing Values by Pandas as NaN are:

1. NA
2. NaN
3. n/a
4. Blank

Check Individual Columns for Null Values

There may be some missing values which are not identified by Pandas as Missing Values (NaN)

In [217]:
```
print (data['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
```

In [218]:
```
print (data['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
```

In [219]:
```
print (data['NUM_BATH'].isnull())
```

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

In [220]:
```
print (data['NUM_BEDROOMS'])
```

```
0      3
1      3
2    NaN
3      1
4      3
5    NaN
6      2
7      1
8     na
Name: NUM_BEDROOMS, dtype: object
```

In [221]:
```
print (data['NUM_BEDROOMS'].isnull())
```

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

In [222]:
```
data['NUM_BEDROOMS'].isnull().sum()
```

Out[222]: 2

4. Configure Missing Values at Read Time

In [223]:
```
missing=["na","--"]
```

In [224]:
```
data=pd.read_csv("D:/FTI/Cohort 2 EDA/Lecture 2/property.csv", na_values=missing)
```

In [225]:
```
data.isnull().sum()
```

Out[225]:
```
PID             1
ST_NUM          2
ST_NAME         0
OWN_OCCUPIED    1
NUM_BEDROOMS    3
NUM_BATH        1
SQ_FT           2
dtype: int64
```

In [226]:
```
print (data['NUM_BEDROOMS'].isnull())
```

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

In [227]:
```
data['NUM_BEDROOMS'].isnull().sum()
```

Out[227]: 3

In [228]:
```
data
```

Out[228]:

	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

5. Invalid data is also treated as missing values , so replace Invalid data with NaN

OWN_OCCUPIED is meant to contain Y and N values . Any other value can be considered as a wrong value .

In [229]:
```
List = ['Y','N',np.nan]
```

In [230]:
```
for row in data['OWN_OCCUPIED']:
    if row in List:
        print("data is valid")
    else:
        print("data is invalid")
```

```
data is valid
data is valid
data is valid
data is invalid
data is valid
data is valid
data is valid
data is valid
data is valid
```

In [ ]:
```
List=['Y','N',np.nan]
```

In [231]:
```
cnt=0
for row in data['OWN_OCCUPIED']:
    if row in List:
        cnt+=1
        pass
    else:
        data.loc[cnt,'OWN_OCCUPIED']=np.nan
        cnt+=1
```

In [232]:
```
data
```

Out[232]:

	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	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

In [233]:
```
print (data.isnull().sum())
```

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

Handling Missing Values

1. Sometimes you may need to replace individual values :

In [234]:
```
data.loc[4,'PID'] = 100005000
```

In [235]:
```
print (data.isnull().sum())
```

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

2. Impute Missing Values Using Mean / Median / Mode

In [236]:
```
median = data['NUM_BEDROOMS'].median()
data['NUM_BEDROOMS'].fillna(median, inplace=True)
```

In [237]:
```
print (data.isnull().sum())
```

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

In [238]:
```
mode = data['NUM_BATH'].mode()
data['NUM_BATH'].fillna(mode, inplace=True)
```

In [239]:
```
print(data.isnull().sum())
```

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

3. Sometimes you may need to delete all rows with missing values

In [240]:
```
data.dropna(inplace=True)
data.reset_index(drop=True, inplace=True)
```

In [241]:
```
data
```

Out[241]:

	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	100005000.0	203.0	BERKELEY	Y	3.0	2	1600.0
2	100006000.0	207.0	BERKELEY	Y	2.5	1	800.0
3	100009000.0	215.0	TREMONT	Y	2.5	2	1800.0

Missing Values PART- 2 (Working with a real world dataset)

In [243]:
```
df=pd.read_csv("D:/FTI/Cohort 2 EDA/Lecture 2/diabetes.csv")
```

In [244]:
```
df.shape
```

Out[244]:
```
(768, 9)
```

In [245]:
```
df.head()
```

Out[245]:

	times_preg	gluco_concent	Diastolic BP	Triceps_Thickness	Hour_insulin	BMI	D_pedigree	Age	Class
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

In [246]:
```
df.columns
```

Out[246]:
```
Index(['times_preg', 'gluco_concent', 'Diastolic BP', 'Triceps_Thickness',
       'Hour_insulin', 'BMI', 'D_pedigree', 'Age ', 'Class'],
      dtype='object')
```

	A	B	C	D	E	F	G
1	PID	ST_NUM	ST_NAME	OWN_OCCUPIED	NUM_BEDROOMS	NUM_BATH	SQ_FT
2	100001000	104	PUTNAM	Y	3	1	1000
3	100002000	197	LEXINGTON	N	3	1.5	--
4	100003000		LEXINGTON	N	n/a	1	850
5	100004000	201	BERKELEY	12	1	NaN	700
6		203	BERKELEY	Y	3	2	1600
7	100006000	207	BERKELEY	Y	NA	1	800
8	100007000	NA	WASHINGTON		2	HURLEY	950
9	100008000	213	TREMONT	Y	1	1	
10	100009000	215	TREMONT	Y	na	2	1800

In [247]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   times_preg         768 non-null    int64  
 1   gluco_concent      768 non-null    int64  
 2   Diastolic BP       768 non-null    int64  
 3   Triceps_Thickness  768 non-null    int64  
 4   Hour_insulin       768 non-null    int64  
 5   BMI                768 non-null    float64
 6   D_pedigree         768 non-null    float64
 7   Age                768 non-null    int64  
 8   Class              768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [248]:
```
df.describe()
```

Out[248]:

	times_preg	gluco_concent	Diastolic BP	Triceps_Thickness	Hour_insulin	BMI	D_pedigree	
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.0
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.2
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.7
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.0
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.0
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.0
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.0
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.0

There are five columns where minimum value is zero , but domain knowledge say that it cannot be zero. These fileds are gluco_content , Diastolic BP, Triceps_Thickness and BMI

In [249]:
```
df.head(20)
```

Out[249]:

	times_preg	gluco_concent	Diastolic BP	Triceps_Thickness	Hour_insulin	BMI	D_pedigree	Age	Class
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
13	1	189	60	23	846	30.1	0.398	59	1
14	5	166	72	19	175	25.8	0.587	51	1
15	7	100	0	0	0	30.0	0.484	32	1
16	0	118	84	47	230	45.8	0.551	31	1
17	7	107	74	0	0	29.6	0.254	31	1
18	1	103	30	38	83	43.3	0.183	33	0
19	1	115	70	30	96	34.6	0.529	32	1

In [250]:
```
print((df[['gluco_concent','Diastolic BP','Triceps_Thickness','Hour_insulin','BMI
']] == 0).sum())
```

```
gluco_concent          5
Diastolic BP          35
Triceps_Thickness    227
Hour_insulin         374
BMI                   11
dtype: int64
```

Replacing 0's with Nan

In [251]:
```
df[['gluco_concent','Diastolic BP','Triceps_Thickness','Hour_insulin','BMI']]= df
[['gluco_concent','Diastolic BP','Triceps_Thickness','Hour_insulin','BMI']].replac
e(0,np.NaN)
```

In [252]:
```
df.isnull().sum()
```

Out[252]:
```
times_preg              0
gluco_concent           5
Diastolic BP           35
Triceps_Thickness     227
Hour_insulin          374
BMI                    11
D_pedigree              0
Age                     0
Class                   0
dtype: int64
```

In [253]:
```
df.head(20)
```

Out[253]:

	times_preg	gluco_concent	Diastolic BP	Triceps_Thickness	Hour_insulin	BMI	D_pedigree	Age	Class
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
5	5	116.0	74.0	NaN	NaN	25.6	0.201	30	0
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26	1
7	10	115.0	NaN	NaN	NaN	35.3	0.134	29	0
8	2	197.0	70.0	45.0	543.0	30.5	0.158	53	1
9	8	125.0	96.0	NaN	NaN	NaN	0.232	54	1
10	4	110.0	92.0	NaN	NaN	37.6	0.191	30	0
11	10	168.0	74.0	NaN	NaN	38.0	0.537	34	1
12	10	139.0	80.0	NaN	NaN	27.1	1.441	57	0
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
15	7	100.0	NaN	NaN	NaN	30.0	0.484	32	1
16	0	118.0	84.0	47.0	230.0	45.8	0.551	31	1
17	7	107.0	74.0	NaN	NaN	29.6	0.254	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

In [254]:
```
df2=df.copy()
```

In [255]:
```
df.head()
```

Out[255]:

	times_preg	gluco_concent	Diastolic BP	Triceps_Thickness	Hour_insulin	BMI	D_pedigree	Age	Class
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

Strategy 01 : Remove Rows or Columns with Missing Data

In [256]:
```
df.dropna(inplace=True)
```

In [257]:
```
df.shape
```

Out[257]: (392, 9)

In [258]:
```
df2.shape
```

Out[258]: (768, 9)

Strategy 02: Replace with Mean Value

In [259]:
```
df2.fillna(df2.mean(), inplace=True)
```

In [260]:
```
df2.head(10)
```

Out[260]:

	times_preg	gluco_concent	Diastolic BP	Triceps_Thickness	Hour_insulin	BMI	D_pedigree	Age	Class
0	6	148.0	72.000000	35.00000	155.548223	33.600000	0.627	50	1
1	1	85.0	66.000000	29.00000	155.548223	26.600000	0.351	31	0
2	8	183.0	64.000000	29.15342	155.548223	23.300000	0.672	32	1
3	1	89.0	66.000000	23.00000	94.000000	28.100000	0.167	21	0
4	0	137.0	40.000000	35.00000	168.000000	43.100000	2.288	33	1
5	5	116.0	74.000000	29.15342	155.548223	25.600000	0.201	30	0
6	3	78.0	50.000000	32.00000	88.000000	31.000000	0.248	26	1
7	10	115.0	72.405184	29.15342	155.548223	35.300000	0.134	29	0
8	2	197.0	70.000000	45.00000	543.000000	30.500000	0.158	53	1
9	8	125.0	96.000000	29.15342	155.548223	32.457464	0.232	54	1

In [261]:
```
df2.isnull().sum()
```

Out[261]:
```
times_preg           0
gluco_concent        0
Diastolic BP         0
Triceps_Thickness    0
Hour_insulin         0
BMI                  0
D_pedigree           0
Age                  0
Class                0
dtype: int64
```

In [263]:
```
df2.to_csv("D:/FTI/Cohort 2 EDA/Lecture 2/diabetes_cleaned.csv")
```