pandas

Data Cleaning

BEGINNER'S CODE GUIDE

Pandas - Cleaning Data

```
Fixing bad data in data set. It could be:
-Empty cells
-Data in wrong format
-Wrong data
-Duplicates
```

Cleaning Empty Cells

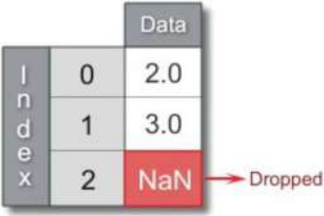
Remove Rows

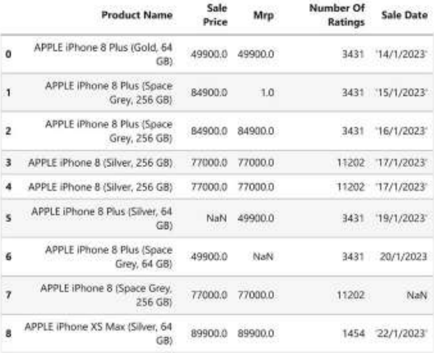
```python
import pandas as pd
```

```python
df = pd.read_csv('data.CSV')
```

```python
df
```

dropna( )

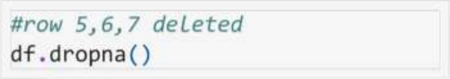




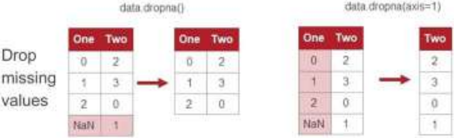
	Product Name	Sale Price	Mrp	Number Of Ratings	Sale Date
0	APPLE iPhone 8 Plus (Gold, 64 GB)	49900.0	49900.0	3431	'14/1/2023'
1	APPLE iPhone 8 Plus (Space Grey, 256 GB)	84900.0	1.0	3431	'15/1/2023'
2	APPLE iPhone 8 Plus (Space Grey, 256 GB)	84900.0	84900.0	3431	'16/1/2023'
3	APPLE iPhone 8 (Silver, 256 GB)	77000.0	77000.0	11202	'17/1/2023'
4	APPLE iPhone 8 (Silver, 256 GB)	77000.0	77000.0	11202	'17/1/2023'
5	APPLE iPhone 8 Plus (Silver, 64 GB)	NaN	49900.0	3431	'19/1/2023'
6	APPLE iPhone 8 Plus (Space Grey, 64 GB)	49900.0	NaN	3431	20/1/2023
7	APPLE iPhone 8 (Space Grey, 256 GB)	77000.0	77000.0	11202	NaN
8	APPLE iPhone XS Max (Silver, 64 GB)	89900.0	89900.0	1454	'22/1/2023'

dropna()

```
remove rows that contain empty cells / null values
```



```
#row 5,6,7 deleted
df.dropna()
```



Drop missing values

data.dropna()

One	Two
0	2
1	3
2	0
NaN	1

One	Two
0	2
1	3
2	0

data.dropna(axis=1)

One	Two
0	2
1	3
2	0
NaN	1

Two
2
3
0
1

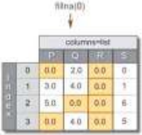
	Product Name	Sale Price	Mrp	Number Of Ratings	Sale Date
0	APPLE iPhone 8 Plus (Gold, 64 GB)	49900.0	49900.0	3431	'14/1/2023'
1	APPLE iPhone 8 Plus (Space Grey, 256 GB)	84900.0	1.0	3431	'15/1/2023'
2	APPLE iPhone 8 Plus (Space Grey, 256 GB)	84900.0	84900.0	3431	'16/1/2023'
3	APPLE iPhone 8 (Silver, 256 GB)	77000.0	77000.0	11202	'17/1/2023'
4	APPLE iPhone 8 (Silver, 256 GB)	77000.0	77000.0	11202	'17/1/2023'
8	APPLE iPhone XS Max (Silver, 64 GB)	89900.0	89900.0	1454	'22/1/2023'

inplace = True

```
By default, the dropna() method returns a new DataFrame,
and will not change the original.
If you want to change the original DataFrame, use the
inplace = True argument
```

```
df.dropna(inplace=True)
```

Replace Empty Values

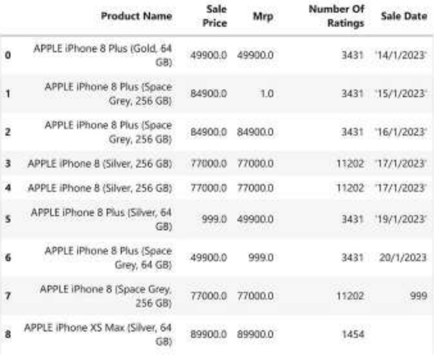


	P	Q	R	S
0	0.0	2.0	0.0	0
1	3.0	4.0	0.0	1
2	5.0	0.0	0.0	6
3	0.0	4.0	0.0	5

fillna() →

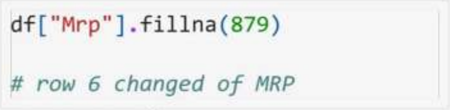
```
replace empty cells with a new value
```

```
#row 5,6,7 null value change to 99

df.fillna(999)
```

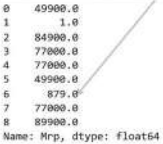


	Product Name	Sale Price	Mrp	Number Of Ratings	Sale Date
0	APPLE iPhone 8 Plus (Gold, 64 GB)	49900.0	49900.0	3431	'14/1/2023'
1	APPLE iPhone 8 Plus (Space Grey, 256 GB)	84900.0	1.0	3431	'15/1/2023'
2	APPLE iPhone 8 Plus (Space Grey, 256 GB)	84900.0	84900.0	3431	'16/1/2023'
3	APPLE iPhone 8 (Silver, 256 GB)	77000.0	77000.0	11202	'17/1/2023'
4	APPLE iPhone 8 (Silver, 256 GB)	77000.0	77000.0	11202	'17/1/2023'
5	APPLE iPhone 8 Plus (Silver, 64 GB)	999.0	49900.0	3431	'19/1/2023'
6	APPLE iPhone 8 Plus (Space Grey, 64 GB)	49900.0	999.0	3431	20/1/2023
7	APPLE iPhone 8 (Space Grey, 256 GB)	77000.0	77000.0	11202	999
8	APPLE iPhone XS Max (Silver, 64 GB)	89900.0	89900.0	1454	

Replace Only For Specified Columns



```
df["Mrp"].fillna(879)

# row 6 changed of MRP
```

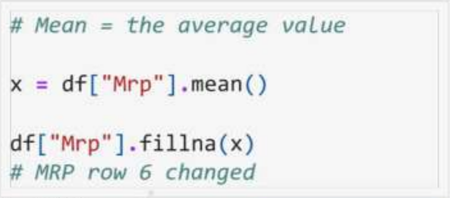


```
0    49900.0
1        1.0
2    84900.0
3    77000.0
4    77000.0
5    49900.0
6      879.0
7    77000.0
8    89900.0
Name: Mrp, dtype: float64
```

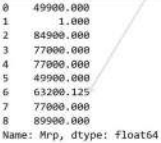
Replace Using Mean, Median, or Mode

calculate the respective values for a specified column

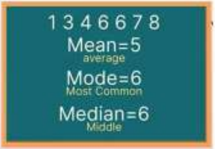
e.g. calculate MEAN, and replace any empty values with it



```
# Mean = the average value

x = df["Mrp"].mean()

df["Mrp"].fillna(x)
# MRP row 6 changed
```



```
0    49900.000
1        1.000
2    84900.000
3    77000.000
4    77000.000
5    49900.000
6    63200.125
7    77000.000
8    89900.000
Name: Mrp, dtype: float64
```



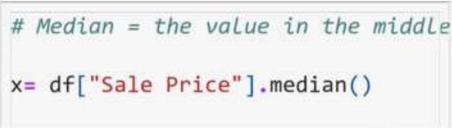
1 3 4 6 6 7 8

Mean=5
average

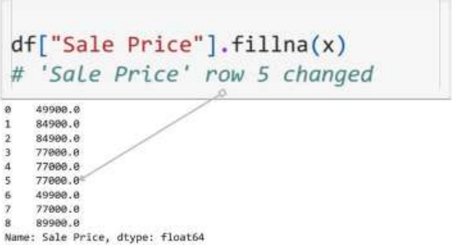
Mode=6
Most Common

Median=6
Middle

calculate the MEDIAN, and replace any empty values with it



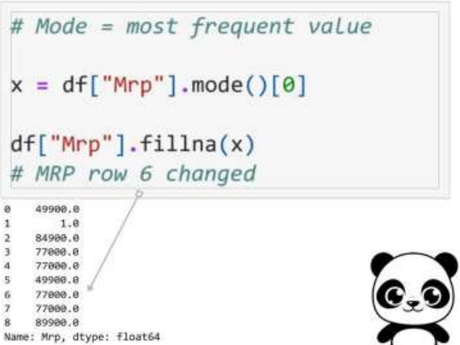
```
# Median = the value in the middle

x= df["Sale Price"].median()
```



```
df["Sale Price"].fillna(x)
# 'Sale Price' row 5 changed
```

```
0    49900.0
1    84900.0
2    84900.0
3    77000.0
4    77000.0
5    77000.0
6    49900.0
7    77000.0
8    89900.0
Name: Sale Price, dtype: float64
```

calculate the MODE, and replace any empty values with it



```
# Mode = most frequent value

x = df["Mrp"].mode()[0]

df["Mrp"].fillna(x)
# MRP row 6 changed
```

```
0    49900.0
1        1.0
2    84900.0
3    77000.0
4    77000.0
5    49900.0
6    77000.0
7    77000.0
8    89900.0
Name: Mrp, dtype: float64
```

Cleaning Data of Wrong Format

```
# Non-date format column

df['Sale Date']
```

```
0    '14/1/2023'
1    '15/1/2023'
2    '16/1/2023'
3    '17/1/2023'
4    '17/1/2023'
5    '19/1/2023'
6      20/1/2023
7            NaN
8    '22/1/2023'
Name: Sale Date, dtype: object
```

Convert Into a Correct Format

to_datetime()

```
df['Sale Date'] =
pd.to_datetime(df['Sale Date'])
```

```
# column in date format
#NaT (Not a Time) i.e. empty cell

df['Sale Date']
```

```
0    2023-01-14
1    2023-01-15
2    2023-01-16
3    2023-01-17
4    2023-01-17
5    2023-01-19
6    2023-01-20
7           NaT
8    2023-01-22
Name: Sale Date, dtype: datetime64[ns]
```

Fixing Wrong Data

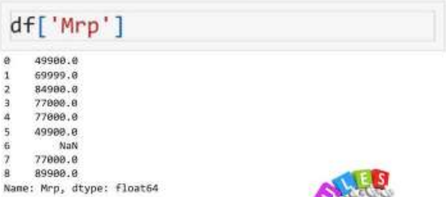
Two way - replace or remove

Replacing Values

```
# Mrp 2nd row incorrect value = 1

df['Mrp']
```

```
0    49900.0
1        1.0
2    84900.0
3    77000.0
4    77000.0
5    49900.0
6        NaN
7    77000.0
8    89900.0
Name: Mrp, dtype: float64
```

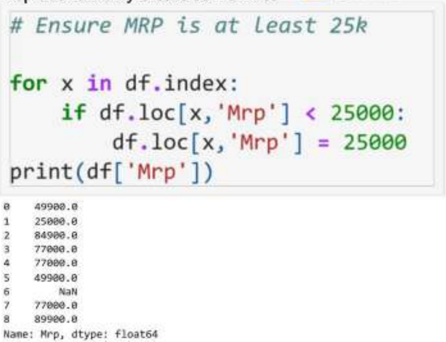
```
# change the value of 2nd row

df.loc[1, 'Mrp'] = 69999
```



```
df['Mrp']
```

```
0    49900.0
1    69999.0
2    84900.0
3    77000.0
4    77000.0
5    49900.0
6        NaN
7    77000.0
8    89900.0
Name: Mrp, dtype: float64
```

Replace value by create some rules



```
# Ensure MRP is at least 25k

for x in df.index:
    if df.loc[x,'Mrp'] < 25000:
        df.loc[x,'Mrp'] = 25000
print(df['Mrp'])
```

```
0    49900.0
1    25000.0
2    84900.0
3    77000.0
4    77000.0
5    49900.0
6        NaN
7    77000.0
8    89900.0
Name: Mrp, dtype: float64
```

Removing Rows of wrong data