pandas

Data Cleaning

BEGINNER'S CODE GUIDE

Ehsan Paydar

https://github.com/EPSOFT

EPSOFT
Everywhere
epsoft98@gmail.com
www.epsoft.ir
@epsoft98
in/ehsanpaydar
epsoft
epsoft98

Pandas - Cleaning Data

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

Cleaning Empty Cells

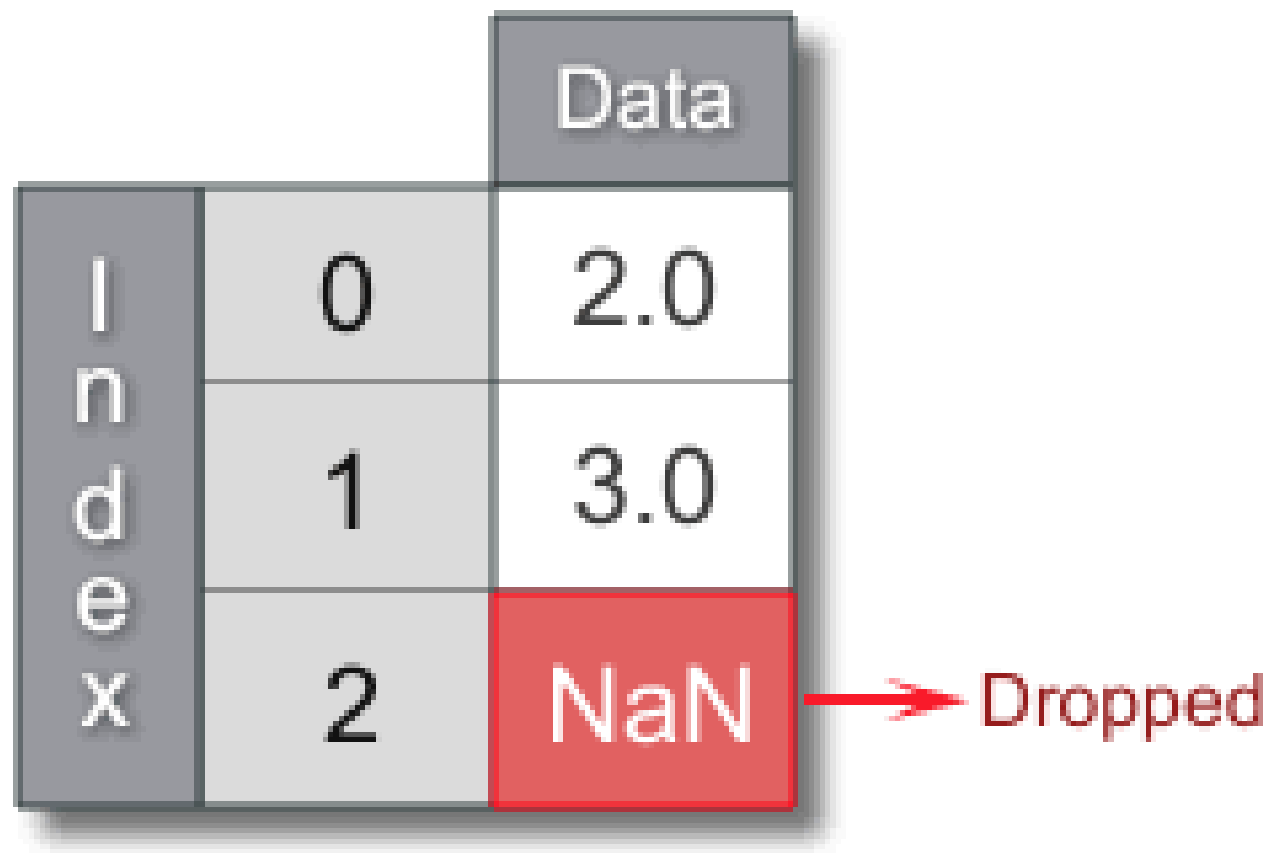
Remove Rows

```
import pandas as pd
```

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

```
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

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
```

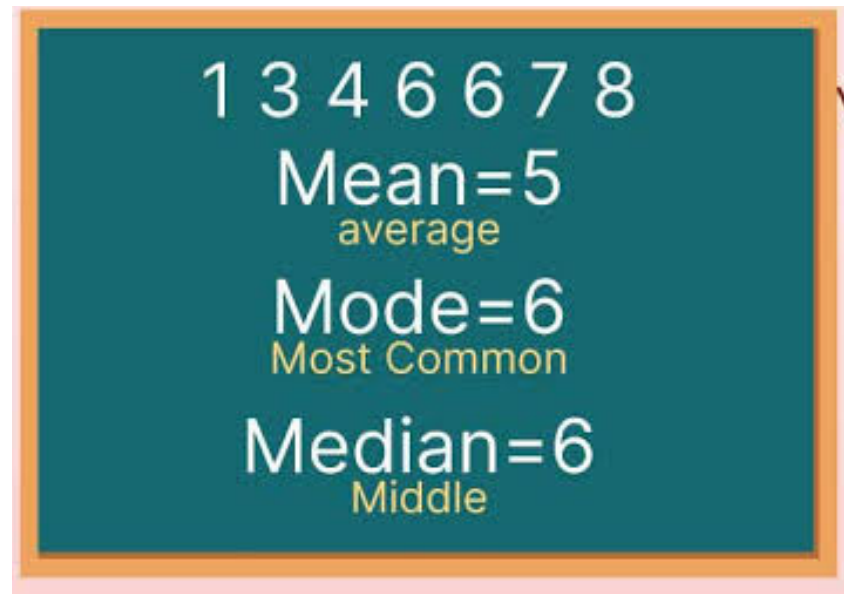
Replace Using Mean, Median, or Mode

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
```



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
```

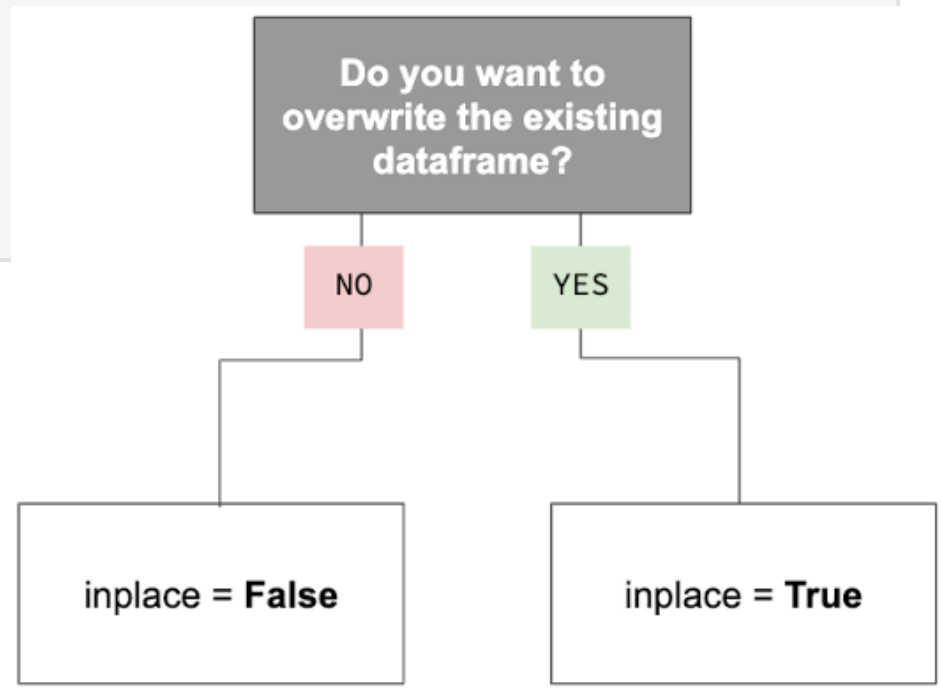
Replace value by create some rules


```
# 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

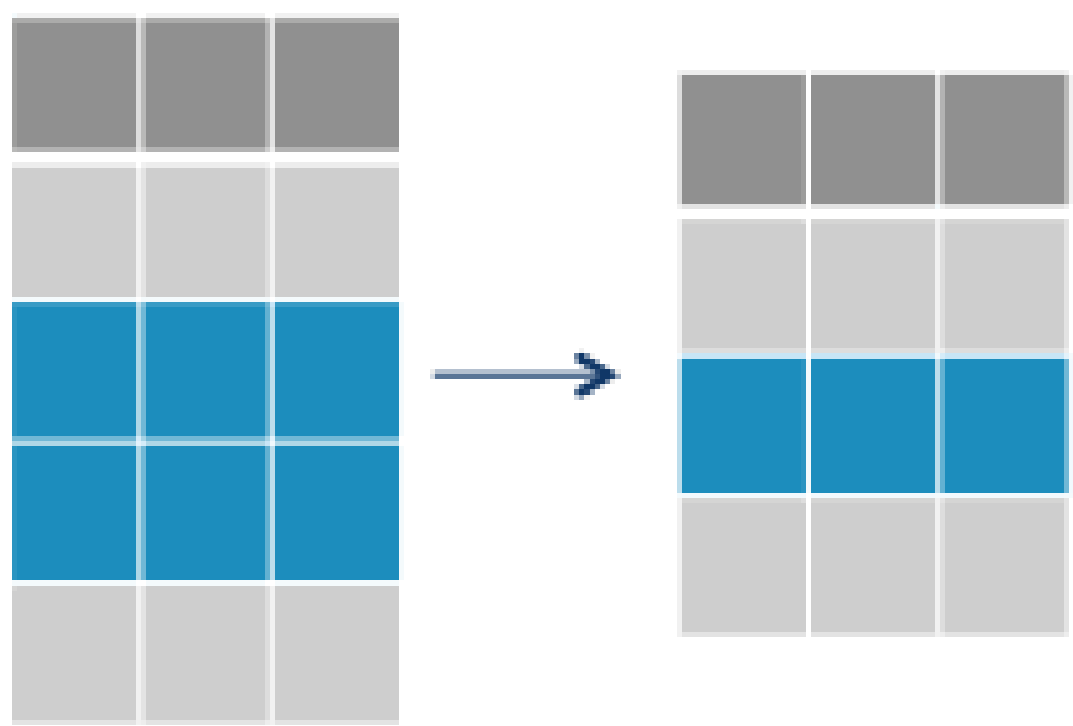
```python
for x in df.index:
    if df.loc[x,'Mrp'] < 25000:
        df.drop(x, inplace=True)

#'inplace=True':modify original DF

print(df['Mrp'])
```




```
0    49900.0
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 Duplicates

```python
df # row 3 & 4 duplicate
```



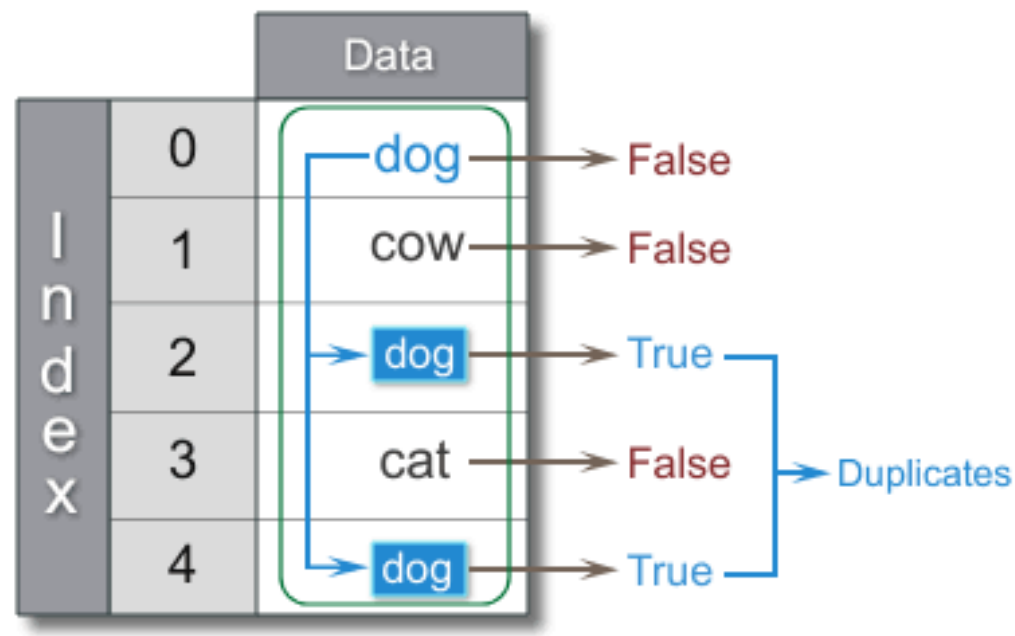
	Product Name	Sale Price	Mrp	Number Of Ratings	Sale Date
0	APPLE iPhone 8 Plus (Gold, 64 GB)	49900.0	49900.0	3431	'14/1/2023'
2	APPLE iPhone 8 Plus (Space Grey, 256 GB)	84900.0	84900.0	3431	'16/1/2023'
3	APPLE iPhone 8 (Silver, 256 GB)	77000.0	77000.0	11202	'17/1/2023'
4	APPLE iPhone 8 (Silver, 256 GB)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

duplicated()

to find duplicate row

```
# row 4 duplicate = True

print(df.duplicated())
```

```
0    False
2    False
3    False
4     True
5    False
6    False
7    False
8    False
dtype: bool
```



drop_duplicates()

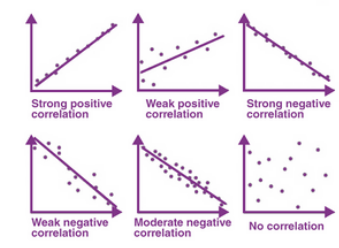
To remove duplicates

```
# delete row 4

df.drop_duplicates()
```

	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'
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'
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'

Data Correlations | corr()

Finding Relationships between column in your data set
The number varies from -1 to 1
1 means that there is a 1 to 1 relationship (a perfect correlation)



```
df[['Sale Price',
    'Mrp',
    'Number Of Ratings']].corr()
```

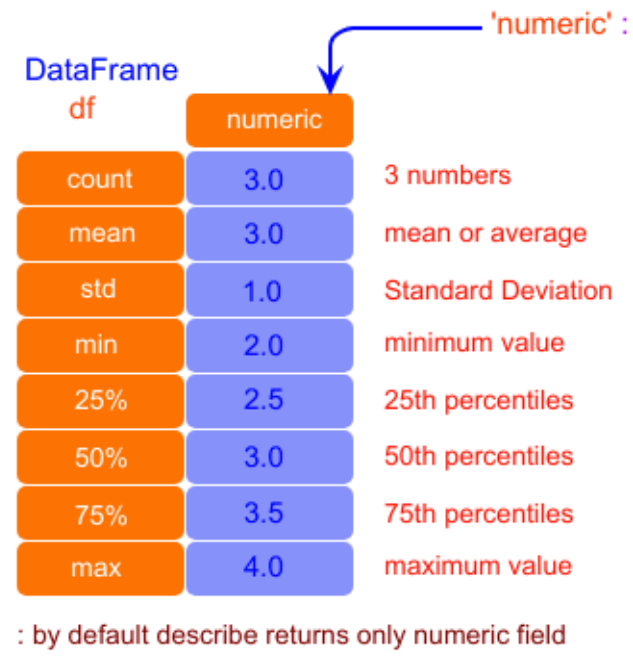
	Sale Price	Mrp	Number Of Ratings
Sale Price	1.000000	1.000000	0.177558
Mrp	1.000000	1.000000	0.177558
Number Of Ratings	0.177558	0.177558	1.000000

describe()

to view some basic statistical details like percentile, mean, std, etc. of a data frame or a series of numeric values

```
df.describe()
```

	Sale Price	Mrp	Number Of Ratings
count	7.000000	7.000000	8.000000
mean	72228.571429	72228.571429	6098.000000
std	16011.840857	16011.840857	4279.030932
min	49900.000000	49900.000000	1454.000000
25%	63450.000000	63450.000000	3431.000000
50%	77000.000000	77000.000000	3431.000000
75%	80950.000000	80950.000000	11202.000000
max	89900.000000	89900.000000	11202.000000



Adding new Column in Dataframe

```
# DF[New Column Name]=Value

df['col'] = 5
```

```
df
```

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

Column Deletion in Dataframe

drop()

```
df.drop(['col','Number Of Ratings'
         axis=1,inplace = True)
```

```
df
```

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

Row Addition

using concat()

```
a = pd.DataFrame({'Product Name':5
                  'Sale Price':6,
                  'Mrp':7,
                  'Sale Date':8},
```

```
                                index =[0])


a
```

	Product Name	Sale Price	Mrp	Sale Date
0	5	6	7	8

```
df = pd.concat([a, df])
```

```
df
```

	Product Name	Sale Price	Mrp	Sale Date
0	5	6.0	7.0	8
1	APPLE iPhone 8 Plus (Space Grey, 256 GB)	84900.0	1.0	'15/1/2023'
2	APPLE iPhone 8 Plus (Space Grey, 256 GB)	84900.0	84900.0	'16/1/2023'
3	APPLE iPhone 8 (Silver, 256 GB)	77000.0	77000.0	'17/1/2023'
4	APPLE iPhone 8 (Silver, 256 GB)	77000.0	77000.0	'17/1/2023'
5	APPLE iPhone 8 Plus (Silver, 64 GB)	NaN	49900.0	'19/1/2023'
6	APPLE iPhone 8 Plus (Space Grey, 64 GB)	49900.0	NaN	20/1/2023
7	APPLE iPhone 8 (Space Grey, 256 GB)	77000.0	77000.0	NaN
8	APPLE iPhone XS Max (Silver, 64 GB)	89900.0	89900.0	'22/1/2023'

Row Deletion in DataFrame

```
df.drop(0,inplace = True)
# index 0 deleted
```

```
df
```

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

Slicing in Pandas

```
df[2:5]    # Start : STOP
```

	Product Name	Sale Price	Mrp	Sale Date
3	APPLE iPhone 8 (Silver, 256 GB)	77000.0	77000.0	'17/1/2023'
4	APPLE iPhone 8 (Silver, 256 GB)	77000.0	77000.0	'17/1/2023'
5	APPLE iPhone 8 Plus (Silver, 64 GB)	NaN	49900.0	'19/1/2023'

```
df[1:7:2]    # Start : STOP : Step
```

	Product Name	Sale Price	Mrp	Sale Date
2	APPLE iPhone 8 Plus (Space Grey, 256 GB)	84900.0	84900.0	'16/1/2023'
4	APPLE iPhone 8 (Silver, 256 GB)	77000.0	77000.0	'17/1/2023'
6	APPLE iPhone 8 Plus (Space Grey, 64 GB)	49900.0	NaN	20/1/2023

```
df[['Mrp']][4:11]    # Start : STOP
```

	Mrp
5	49900.0
6	NaN
7	77000.0
8	89900.0

Thank you!