!pip install pandas

Requirement already satisfied: pandas in c:\users\jai\appdata\local\programs\python\python312\lib\site-packages (2.2.1)

[notice] A new release of pip is available: 24.1.2 -> 24.2  
[notice] To update, run: python.exe -m pip install --upgrade pip

Requirement already satisfied: numpy<2,>=1.26.0 in c:\users\jai\appdata\local\programs\python\python312\lib\site-packages (from pandas) (1.26.4)  
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\jai\appdata\local\programs\python\python312\lib\site-packages (from pandas) (2.8.2)  
Requirement already satisfied: pytz>=2020.1 in c:\users\jai\appdata\local\programs\python\python312\lib\site-packages (from pandas) (2024.1)  
Requirement already satisfied: tzdata>=2022.7 in c:\users\jai\appdata\local\programs\python\python312\lib\site-packages (from pandas) (2024.1)  
Requirement already satisfied: six>=1.5 in c:\users\jai\appdata\local\programs\python\python312\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)

import pandas as pd

df = pd.read\_csv('Used\_Bikes.csv')  
df

bike\_name price city kms\_driven \  
0 TVS Star City Plus Dual Tone 110cc 35000.0 Ahmedabad 17654.0   
1 Royal Enfield Classic 350cc 119900.0 Delhi 11000.0   
2 Triumph Daytona 675R 600000.0 Delhi 110.0   
3 TVS Apache RTR 180cc 65000.0 Bangalore 16329.0   
4 Yamaha FZ S V 2.0 150cc-Ltd. Edition 80000.0 Bangalore 10000.0   
... ... ... ... ...   
32643 Hero Passion Pro 100cc 39000.0 Delhi 22000.0   
32644 TVS Apache RTR 180cc 30000.0 Karnal 6639.0   
32645 Bajaj Avenger Street 220 60000.0 Delhi 20373.0   
32646 Hero Super Splendor 125cc 15600.0 Jaipur 84186.0   
32647 Bajaj Pulsar 150cc 22000.0 Pune 60857.0   
  
 owner age power brand   
0 First Owner 3.0 110.0 TVS   
1 First Owner 4.0 350.0 Royal Enfield   
2 First Owner 8.0 675.0 Triumph   
3 First Owner 4.0 180.0 TVS   
4 First Owner 3.0 150.0 Yamaha   
... ... ... ... ...   
32643 First Owner 4.0 100.0 Hero   
32644 First Owner 9.0 180.0 TVS   
32645 First Owner 6.0 220.0 Bajaj   
32646 First Owner 16.0 125.0 Hero   
32647 First Owner 13.0 150.0 Bajaj   
  
[32648 rows x 8 columns]

type(df)

pandas.core.frame.DataFrame

!pip install openpyxl

Requirement already satisfied: openpyxl in c:\users\jai\appdata\local\programs\python\python312\lib\site-packages (3.1.2)  
Requirement already satisfied: et-xmlfile in c:\users\jai\appdata\local\programs\python\python312\lib\site-packages (from openpyxl) (1.1.0)

[notice] A new release of pip is available: 24.1.2 -> 24.2  
[notice] To update, run: python.exe -m pip install --upgrade pip

pd.read\_excel('data.xlsx')

bike\_name price city kms\_driven \  
0 TVS Star City Plus Dual Tone 110cc 35000 Ahmedabad 17654   
1 Royal Enfield Classic 350cc 119900 Delhi 11000   
2 Triumph Daytona 675R 600000 Delhi 110   
3 TVS Apache RTR 180cc 65000 Bangalore 16329   
4 Yamaha FZ S V 2.0 150cc-Ltd. Edition 80000 Bangalore 10000   
... ... ... ... ...   
32643 Hero Passion Pro 100cc 39000 Delhi 22000   
32644 TVS Apache RTR 180cc 30000 Karnal 6639   
32645 Bajaj Avenger Street 220 60000 Delhi 20373   
32646 Hero Super Splendor 125cc 15600 Jaipur 84186   
32647 Bajaj Pulsar 150cc 22000 Pune 60857   
  
 owner age power brand   
0 First Owner 3 110 TVS   
1 First Owner 4 350 Royal Enfield   
2 First Owner 8 675 Triumph   
3 First Owner 4 180 TVS   
4 First Owner 3 150 Yamaha   
... ... ... ... ...   
32643 First Owner 4 100 Hero   
32644 First Owner 9 180 TVS   
32645 First Owner 6 220 Bajaj   
32646 First Owner 16 125 Hero   
32647 First Owner 13 150 Bajaj   
  
[32648 rows x 8 columns]

dt = {'name':['rohit','mohit','rahul'],'branch':['cse','it','ece'],'class':[1,2,3]}  
dt

{'name': ['rohit', 'mohit', 'rahul'],  
 'branch': ['cse', 'it', 'ece'],  
 'class': [1, 2, 3]}

pd.DataFrame(dt)

name branch class  
0 rohit cse 1  
1 mohit it 2  
2 rahul ece 3

lst = list(range(1,11))  
lst

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

ser1 = pd.Series(lst,name='serial\_no')  
ser1

0 1  
1 2  
2 3  
3 4  
4 5  
5 6  
6 7  
7 8  
8 9  
9 10  
Name: serial\_no, dtype: int64

ser2 = pd.Series(lst,name='series\_2')  
ser2

0 1  
1 2  
2 3  
3 4  
4 5  
5 6  
6 7  
7 8  
8 9  
9 10  
Name: series\_2, dtype: int64

pd.concat([ser1,ser2],axis='columns')

serial\_no series\_2  
0 1 1  
1 2 2  
2 3 3  
3 4 4  
4 5 5  
5 6 6  
6 7 7  
7 8 8  
8 9 9  
9 10 10

df2 = pd.concat([ser1,ser2],axis=1)  
df2

serial\_no series\_2  
0 1 1  
1 2 2  
2 3 3  
3 4 4  
4 5 5  
5 6 6  
6 7 7  
7 8 8  
8 9 9  
9 10 10

df2.shape

(10, 2)

import numpy as np

arr = np.arange(1,61).reshape(12,5)  
arr

array([[ 1, 2, 3, 4, 5],  
 [ 6, 7, 8, 9, 10],  
 [11, 12, 13, 14, 15],  
 [16, 17, 18, 19, 20],  
 [21, 22, 23, 24, 25],  
 [26, 27, 28, 29, 30],  
 [31, 32, 33, 34, 35],  
 [36, 37, 38, 39, 40],  
 [41, 42, 43, 44, 45],  
 [46, 47, 48, 49, 50],  
 [51, 52, 53, 54, 55],  
 [56, 57, 58, 59, 60]])

pd.DataFrame(arr)

0 1 2 3 4  
0 1 2 3 4 5  
1 6 7 8 9 10  
2 11 12 13 14 15  
3 16 17 18 19 20  
4 21 22 23 24 25  
5 26 27 28 29 30  
6 31 32 33 34 35  
7 36 37 38 39 40  
8 41 42 43 44 45  
9 46 47 48 49 50  
10 51 52 53 54 55  
11 56 57 58 59 60

pd.DataFrame(arr,columns=['col1','col2','col3','col4','col5'])

col1 col2 col3 col4 col5  
0 1 2 3 4 5  
1 6 7 8 9 10  
2 11 12 13 14 15  
3 16 17 18 19 20  
4 21 22 23 24 25  
5 26 27 28 29 30  
6 31 32 33 34 35  
7 36 37 38 39 40  
8 41 42 43 44 45  
9 46 47 48 49 50  
10 51 52 53 54 55  
11 56 57 58 59 60

pd.DataFrame(np.arange(1,61).reshape((12,5)),columns=['col1','col2','col3','col4','col5'])

col1 col2 col3 col4 col5  
0 1 2 3 4 5  
1 6 7 8 9 10  
2 11 12 13 14 15  
3 16 17 18 19 20  
4 21 22 23 24 25  
5 26 27 28 29 30  
6 31 32 33 34 35  
7 36 37 38 39 40  
8 41 42 43 44 45  
9 46 47 48 49 50  
10 51 52 53 54 55  
11 56 57 58 59 60

arr = np.random.randint(1,200,100)  
arr

array([ 3, 170, 7, 77, 155, 26, 178, 115, 106, 156, 59, 146, 87,  
 106, 18, 184, 59, 177, 83, 149, 70, 142, 33, 31, 66, 57,  
 194, 7, 194, 77, 164, 174, 94, 106, 130, 129, 56, 141, 74,  
 183, 51, 189, 16, 25, 3, 71, 70, 123, 94, 57, 172, 83,  
 123, 154, 199, 133, 80, 57, 65, 107, 58, 186, 52, 16, 86,  
 64, 129, 54, 65, 161, 13, 174, 91, 36, 21, 187, 106, 91,  
 65, 149, 145, 125, 78, 163, 123, 51, 192, 184, 40, 161, 173,  
 177, 105, 123, 150, 122, 191, 99, 117, 169])

ser = pd.Series(arr,name='serial\_no')  
ser

0 3  
1 170  
2 7  
3 77  
4 155  
 ...   
95 122  
96 191  
97 99  
98 117  
99 169  
Name: serial\_no, Length: 100, dtype: int32

ser[0:4]

0 3  
1 170  
2 7  
3 77  
Name: serial\_no, dtype: int32

arr[arr>100]

array([170, 155, 178, 115, 106, 156, 146, 106, 184, 177, 149, 142, 194,  
 194, 164, 174, 106, 130, 129, 141, 183, 189, 123, 172, 123, 154,  
 199, 133, 107, 186, 129, 161, 174, 187, 106, 149, 145, 125, 163,  
 123, 192, 184, 161, 173, 177, 105, 123, 150, 122, 191, 117, 169])

ser[ser <197]

0 3  
1 170  
2 7  
3 77  
4 155  
 ...   
95 122  
96 191  
97 99  
98 117  
99 169  
Name: serial\_no, Length: 99, dtype: int32

type(ser)

pandas.core.series.Series

df

bike\_name price city kms\_driven \  
0 TVS Star City Plus Dual Tone 110cc 35000.0 Ahmedabad 17654.0   
1 Royal Enfield Classic 350cc 119900.0 Delhi 11000.0   
2 Triumph Daytona 675R 600000.0 Delhi 110.0   
3 TVS Apache RTR 180cc 65000.0 Bangalore 16329.0   
4 Yamaha FZ S V 2.0 150cc-Ltd. Edition 80000.0 Bangalore 10000.0   
... ... ... ... ...   
32643 Hero Passion Pro 100cc 39000.0 Delhi 22000.0   
32644 TVS Apache RTR 180cc 30000.0 Karnal 6639.0   
32645 Bajaj Avenger Street 220 60000.0 Delhi 20373.0   
32646 Hero Super Splendor 125cc 15600.0 Jaipur 84186.0   
32647 Bajaj Pulsar 150cc 22000.0 Pune 60857.0   
  
 owner age power brand   
0 First Owner 3.0 110.0 TVS   
1 First Owner 4.0 350.0 Royal Enfield   
2 First Owner 8.0 675.0 Triumph   
3 First Owner 4.0 180.0 TVS   
4 First Owner 3.0 150.0 Yamaha   
... ... ... ... ...   
32643 First Owner 4.0 100.0 Hero   
32644 First Owner 9.0 180.0 TVS   
32645 First Owner 6.0 220.0 Bajaj   
32646 First Owner 16.0 125.0 Hero   
32647 First Owner 13.0 150.0 Bajaj   
  
[32648 rows x 8 columns]

df['brand']

0 TVS  
1 Royal Enfield  
2 Triumph  
3 TVS  
4 Yamaha  
 ...   
32643 Hero  
32644 TVS  
32645 Bajaj  
32646 Hero  
32647 Bajaj  
Name: brand, Length: 32648, dtype: object

type(df['brand'])

pandas.core.series.Series

arr.dtype

dtype('int32')

ser.dtypes

dtype('int32')

df.dtypes

bike\_name object  
price float64  
city object  
kms\_driven float64  
owner object  
age float64  
power float64  
brand object  
dtype: object

df

bike\_name price city kms\_driven \  
0 TVS Star City Plus Dual Tone 110cc 35000.0 Ahmedabad 17654.0   
1 Royal Enfield Classic 350cc 119900.0 Delhi 11000.0   
2 Triumph Daytona 675R 600000.0 Delhi 110.0   
3 TVS Apache RTR 180cc 65000.0 Bangalore 16329.0   
4 Yamaha FZ S V 2.0 150cc-Ltd. Edition 80000.0 Bangalore 10000.0   
... ... ... ... ...   
32643 Hero Passion Pro 100cc 39000.0 Delhi 22000.0   
32644 TVS Apache RTR 180cc 30000.0 Karnal 6639.0   
32645 Bajaj Avenger Street 220 60000.0 Delhi 20373.0   
32646 Hero Super Splendor 125cc 15600.0 Jaipur 84186.0   
32647 Bajaj Pulsar 150cc 22000.0 Pune 60857.0   
  
 owner age power brand   
0 First Owner 3.0 110.0 TVS   
1 First Owner 4.0 350.0 Royal Enfield   
2 First Owner 8.0 675.0 Triumph   
3 First Owner 4.0 180.0 TVS   
4 First Owner 3.0 150.0 Yamaha   
... ... ... ... ...   
32643 First Owner 4.0 100.0 Hero   
32644 First Owner 9.0 180.0 TVS   
32645 First Owner 6.0 220.0 Bajaj   
32646 First Owner 16.0 125.0 Hero   
32647 First Owner 13.0 150.0 Bajaj   
  
[32648 rows x 8 columns]

# df.brand  
df['brand']

0 TVS  
1 Royal Enfield  
2 Triumph  
3 TVS  
4 Yamaha  
 ...   
32643 Hero  
32644 TVS  
32645 Bajaj  
32646 Hero  
32647 Bajaj  
Name: brand, Length: 32648, dtype: object

df['brand']

0 TVS  
1 Royal Enfield  
2 Triumph  
3 TVS  
4 Yamaha  
 ...   
32643 Hero  
32644 TVS  
32645 Bajaj  
32646 Hero  
32647 Bajaj  
Name: brand, Length: 32648, dtype: object

df[['brand','age','price']]

brand age price  
0 TVS 3.0 35000.0  
1 Royal Enfield 4.0 119900.0  
2 Triumph 8.0 600000.0  
3 TVS 4.0 65000.0  
4 Yamaha 3.0 80000.0  
... ... ... ...  
32643 Hero 4.0 39000.0  
32644 TVS 9.0 30000.0  
32645 Bajaj 6.0 60000.0  
32646 Hero 16.0 15600.0  
32647 Bajaj 13.0 22000.0  
  
[32648 rows x 3 columns]

df[0:2]

bike\_name price city kms\_driven \  
0 TVS Star City Plus Dual Tone 110cc 35000.0 Ahmedabad 17654.0   
1 Royal Enfield Classic 350cc 119900.0 Delhi 11000.0   
  
 owner age power brand   
0 First Owner 3.0 110.0 TVS   
1 First Owner 4.0 350.0 Royal Enfield

df2 = df[['brand','age','price']]  
df2[0:50]

brand age price  
0 TVS 3.0 35000.0  
1 Royal Enfield 4.0 119900.0  
2 Triumph 8.0 600000.0  
3 TVS 4.0 65000.0  
4 Yamaha 3.0 80000.0  
5 Yamaha 6.0 53499.0  
6 Honda 3.0 85000.0  
7 Hero 3.0 45000.0  
8 Royal Enfield 3.0 145000.0  
9 Royal Enfield 7.0 88000.0  
10 Yamaha 7.0 72000.0  
11 Yamaha 4.0 95000.0  
12 Bajaj 4.0 78000.0  
13 Bajaj 8.0 29499.0  
14 Bajaj 7.0 29900.0  
15 Bajaj 3.0 90000.0  
16 Bajaj 3.0 120000.0  
17 Suzuki 5.0 48000.0  
18 Benelli 3.0 240000.0  
19 Bajaj 7.0 29900.0  
20 Bajaj 3.0 120000.0  
21 Suzuki 5.0 48000.0  
22 Hero 2.0 46500.0  
23 Royal Enfield 5.0 121700.0  
24 Yamaha 6.0 45000.0  
25 Bajaj 4.0 78000.0  
26 Hero 16.0 20000.0  
27 Honda 7.0 20800.0  
28 Bajaj 8.0 50000.0  
29 Honda 2.0 81200.0  
30 Bajaj 3.0 88000.0  
31 Yamaha 4.0 80000.0  
32 Bajaj 6.0 40900.0  
33 KTM 4.0 180000.0  
34 Honda 8.0 40000.0  
35 KTM 8.0 70000.0  
36 Royal Enfield 5.0 98800.0  
37 Honda 6.0 65000.0  
38 Royal Enfield 2.0 190500.0  
39 KTM 2.0 179000.0  
40 Royal Enfield 8.0 70000.0  
41 Bajaj 6.0 45000.0  
42 Royal Enfield 13.0 78000.0  
43 Bajaj 2.0 110000.0  
44 Royal Enfield 6.0 85000.0  
45 Bajaj 9.0 38000.0  
46 Royal Enfield 10.0 55346.0  
47 Mahindra 2.0 28000.0  
48 Hero 8.0 37000.0  
49 Suzuki 3.0 97000.0

## LOC & ILOC  
df.loc[0:10:2,['age','price',"brand"]]

age price brand  
0 3.0 35000.0 TVS  
2 8.0 600000.0 Triumph  
4 3.0 80000.0 Yamaha  
6 3.0 85000.0 Honda  
8 3.0 145000.0 Royal Enfield  
10 7.0 72000.0 Yamaha

df.iloc[0:10,3:5]

kms\_driven owner  
0 17654.0 First Owner  
1 11000.0 First Owner  
2 110.0 First Owner  
3 16329.0 First Owner  
4 10000.0 First Owner  
5 25000.0 First Owner  
6 8200.0 First Owner  
7 12645.0 First Owner  
8 9190.0 First Owner  
9 19000.0 Second Owner

df

bike\_name price city kms\_driven \  
0 TVS Star City Plus Dual Tone 110cc 35000.0 Ahmedabad 17654.0   
1 Royal Enfield Classic 350cc 119900.0 Delhi 11000.0   
2 Triumph Daytona 675R 600000.0 Delhi 110.0   
3 TVS Apache RTR 180cc 65000.0 Bangalore 16329.0   
4 Yamaha FZ S V 2.0 150cc-Ltd. Edition 80000.0 Bangalore 10000.0   
... ... ... ... ...   
32643 Hero Passion Pro 100cc 39000.0 Delhi 22000.0   
32644 TVS Apache RTR 180cc 30000.0 Karnal 6639.0   
32645 Bajaj Avenger Street 220 60000.0 Delhi 20373.0   
32646 Hero Super Splendor 125cc 15600.0 Jaipur 84186.0   
32647 Bajaj Pulsar 150cc 22000.0 Pune 60857.0   
  
 owner age power brand   
0 First Owner 3.0 110.0 TVS   
1 First Owner 4.0 350.0 Royal Enfield   
2 First Owner 8.0 675.0 Triumph   
3 First Owner 4.0 180.0 TVS   
4 First Owner 3.0 150.0 Yamaha   
... ... ... ... ...   
32643 First Owner 4.0 100.0 Hero   
32644 First Owner 9.0 180.0 TVS   
32645 First Owner 6.0 220.0 Bajaj   
32646 First Owner 16.0 125.0 Hero   
32647 First Owner 13.0 150.0 Bajaj   
  
[32648 rows x 8 columns]

# brand,power,price top 20 rows

df.loc[0:20,['brand','power','price']]

brand power price  
0 TVS 110.0 35000.0  
1 Royal Enfield 350.0 119900.0  
2 Triumph 675.0 600000.0  
3 TVS 180.0 65000.0  
4 Yamaha 150.0 80000.0  
5 Yamaha 150.0 53499.0  
6 Honda 160.0 85000.0  
7 Hero 100.0 45000.0  
8 Royal Enfield 350.0 145000.0  
9 Royal Enfield 500.0 88000.0  
10 Yamaha 150.0 72000.0  
11 Yamaha 250.0 95000.0  
12 Bajaj 200.0 78000.0  
13 Bajaj 100.0 29499.0  
14 Bajaj 125.0 29900.0  
15 Bajaj 200.0 90000.0  
16 Bajaj 200.0 120000.0  
17 Suzuki 150.0 48000.0  
18 Benelli 302.0 240000.0  
19 Bajaj 125.0 29900.0  
20 Bajaj 200.0 120000.0

df2 = df[['brand','price','power']]  
df2[0:51]

brand price power  
0 TVS 35000.0 110.0  
1 Royal Enfield 119900.0 350.0  
2 Triumph 600000.0 675.0  
3 TVS 65000.0 180.0  
4 Yamaha 80000.0 150.0  
5 Yamaha 53499.0 150.0  
6 Honda 85000.0 160.0  
7 Hero 45000.0 100.0  
8 Royal Enfield 145000.0 350.0  
9 Royal Enfield 88000.0 500.0  
10 Yamaha 72000.0 150.0  
11 Yamaha 95000.0 250.0  
12 Bajaj 78000.0 200.0  
13 Bajaj 29499.0 100.0  
14 Bajaj 29900.0 125.0  
15 Bajaj 90000.0 200.0  
16 Bajaj 120000.0 200.0  
17 Suzuki 48000.0 150.0  
18 Benelli 240000.0 302.0  
19 Bajaj 29900.0 125.0  
20 Bajaj 120000.0 200.0  
21 Suzuki 48000.0 150.0  
22 Hero 46500.0 110.0  
23 Royal Enfield 121700.0 500.0  
24 Yamaha 45000.0 150.0  
25 Bajaj 78000.0 200.0  
26 Hero 20000.0 125.0  
27 Honda 20800.0 125.0  
28 Bajaj 50000.0 150.0  
29 Honda 81200.0 160.0  
30 Bajaj 88000.0 200.0  
31 Yamaha 80000.0 150.0  
32 Bajaj 40900.0 220.0  
33 KTM 180000.0 390.0  
34 Honda 40000.0 150.0  
35 KTM 70000.0 200.0  
36 Royal Enfield 98800.0 350.0  
37 Honda 65000.0 150.0  
38 Royal Enfield 190500.0 500.0  
39 KTM 179000.0 200.0  
40 Royal Enfield 70000.0 350.0  
41 Bajaj 45000.0 220.0  
42 Royal Enfield 78000.0 350.0  
43 Bajaj 110000.0 220.0  
44 Royal Enfield 85000.0 350.0  
45 Bajaj 38000.0 200.0  
46 Royal Enfield 55346.0 350.0  
47 Mahindra 28000.0 110.0  
48 Hero 37000.0 150.0  
49 Suzuki 97000.0 150.0  
50 Yamaha 40000.0 150.0

df['price'].min()

4400.0

df[['price']].min()

price 4400.0  
dtype: float64

df.min()

bike\_name BMW F750 GS 850cc  
price 4400.0  
city 24 Pargana  
kms\_driven 1.0  
owner First Owner  
age 1.0  
power 100.0  
brand BMW  
dtype: object

df

bike\_name price city kms\_driven \  
0 TVS Star City Plus Dual Tone 110cc 35000.0 Ahmedabad 17654.0   
1 Royal Enfield Classic 350cc 119900.0 Delhi 11000.0   
2 Triumph Daytona 675R 600000.0 Delhi 110.0   
3 TVS Apache RTR 180cc 65000.0 Bangalore 16329.0   
4 Yamaha FZ S V 2.0 150cc-Ltd. Edition 80000.0 Bangalore 10000.0   
... ... ... ... ...   
32643 Hero Passion Pro 100cc 39000.0 Delhi 22000.0   
32644 TVS Apache RTR 180cc 30000.0 Karnal 6639.0   
32645 Bajaj Avenger Street 220 60000.0 Delhi 20373.0   
32646 Hero Super Splendor 125cc 15600.0 Jaipur 84186.0   
32647 Bajaj Pulsar 150cc 22000.0 Pune 60857.0   
  
 owner age power brand   
0 First Owner 3.0 110.0 TVS   
1 First Owner 4.0 350.0 Royal Enfield   
2 First Owner 8.0 675.0 Triumph   
3 First Owner 4.0 180.0 TVS   
4 First Owner 3.0 150.0 Yamaha   
... ... ... ... ...   
32643 First Owner 4.0 100.0 Hero   
32644 First Owner 9.0 180.0 TVS   
32645 First Owner 6.0 220.0 Bajaj   
32646 First Owner 16.0 125.0 Hero   
32647 First Owner 13.0 150.0 Bajaj   
  
[32648 rows x 8 columns]

# df['price'].min()  
# df['price'].max()  
# df['price'].mean()  
df['owner'].unique()

array(['First Owner', 'Second Owner', 'Third Owner',  
 'Fourth Owner Or More'], dtype=object)

df['owner'].value\_counts()

owner  
First Owner 29964  
Second Owner 2564  
Third Owner 108  
Fourth Owner Or More 12  
Name: count, dtype: int64

df['bike\_name'].nunique()

471

df.head()

bike\_name price city kms\_driven \  
0 TVS Star City Plus Dual Tone 110cc 35000.0 Ahmedabad 17654.0   
1 Royal Enfield Classic 350cc 119900.0 Delhi 11000.0   
2 Triumph Daytona 675R 600000.0 Delhi 110.0   
3 TVS Apache RTR 180cc 65000.0 Bangalore 16329.0   
4 Yamaha FZ S V 2.0 150cc-Ltd. Edition 80000.0 Bangalore 10000.0   
  
 owner age power brand   
0 First Owner 3.0 110.0 TVS   
1 First Owner 4.0 350.0 Royal Enfield   
2 First Owner 8.0 675.0 Triumph   
3 First Owner 4.0 180.0 TVS   
4 First Owner 3.0 150.0 Yamaha

df.tail()

bike\_name price city kms\_driven owner \  
32643 Hero Passion Pro 100cc 39000.0 Delhi 22000.0 First Owner   
32644 TVS Apache RTR 180cc 30000.0 Karnal 6639.0 First Owner   
32645 Bajaj Avenger Street 220 60000.0 Delhi 20373.0 First Owner   
32646 Hero Super Splendor 125cc 15600.0 Jaipur 84186.0 First Owner   
32647 Bajaj Pulsar 150cc 22000.0 Pune 60857.0 First Owner   
  
 age power brand   
32643 4.0 100.0 Hero   
32644 9.0 180.0 TVS   
32645 6.0 220.0 Bajaj   
32646 16.0 125.0 Hero   
32647 13.0 150.0 Bajaj

df.head(10)

bike\_name price city kms\_driven \  
0 TVS Star City Plus Dual Tone 110cc 35000.0 Ahmedabad 17654.0   
1 Royal Enfield Classic 350cc 119900.0 Delhi 11000.0   
2 Triumph Daytona 675R 600000.0 Delhi 110.0   
3 TVS Apache RTR 180cc 65000.0 Bangalore 16329.0   
4 Yamaha FZ S V 2.0 150cc-Ltd. Edition 80000.0 Bangalore 10000.0   
5 Yamaha FZs 150cc 53499.0 Delhi 25000.0   
6 Honda CB Hornet 160R ABS DLX 85000.0 Delhi 8200.0   
7 Hero Splendor Plus Self Alloy 100cc 45000.0 Delhi 12645.0   
8 Royal Enfield Thunderbird X 350cc 145000.0 Bangalore 9190.0   
9 Royal Enfield Classic Desert Storm 500cc 88000.0 Delhi 19000.0   
  
 owner age power brand   
0 First Owner 3.0 110.0 TVS   
1 First Owner 4.0 350.0 Royal Enfield   
2 First Owner 8.0 675.0 Triumph   
3 First Owner 4.0 180.0 TVS   
4 First Owner 3.0 150.0 Yamaha   
5 First Owner 6.0 150.0 Yamaha   
6 First Owner 3.0 160.0 Honda   
7 First Owner 3.0 100.0 Hero   
8 First Owner 3.0 350.0 Royal Enfield   
9 Second Owner 7.0 500.0 Royal Enfield

df.columns

Index(['bike\_name', 'price', 'city', 'kms\_driven', 'owner', 'age', 'power',  
 'brand'],  
 dtype='object')

df.columns

Index(['bike\_name', 'price', 'city', 'kms\_driven', 'owner', 'age', 'power',  
 'brand'],  
 dtype='object')

df[['brand','power']]

brand power  
0 TVS 110.0  
1 Royal Enfield 350.0  
2 Triumph 675.0  
3 TVS 180.0  
4 Yamaha 150.0  
... ... ...  
32643 Hero 100.0  
32644 TVS 180.0  
32645 Bajaj 220.0  
32646 Hero 125.0  
32647 Bajaj 150.0  
  
[32648 rows x 2 columns]

df

bike\_name price city kms\_driven \  
0 TVS Star City Plus Dual Tone 110cc 35000.0 Ahmedabad 17654.0   
1 Royal Enfield Classic 350cc 119900.0 Delhi 11000.0   
2 Triumph Daytona 675R 600000.0 Delhi 110.0   
3 TVS Apache RTR 180cc 65000.0 Bangalore 16329.0   
4 Yamaha FZ S V 2.0 150cc-Ltd. Edition 80000.0 Bangalore 10000.0   
... ... ... ... ...   
32643 Hero Passion Pro 100cc 39000.0 Delhi 22000.0   
32644 TVS Apache RTR 180cc 30000.0 Karnal 6639.0   
32645 Bajaj Avenger Street 220 60000.0 Delhi 20373.0   
32646 Hero Super Splendor 125cc 15600.0 Jaipur 84186.0   
32647 Bajaj Pulsar 150cc 22000.0 Pune 60857.0   
  
 owner age power brand   
0 First Owner 3.0 110.0 TVS   
1 First Owner 4.0 350.0 Royal Enfield   
2 First Owner 8.0 675.0 Triumph   
3 First Owner 4.0 180.0 TVS   
4 First Owner 3.0 150.0 Yamaha   
... ... ... ... ...   
32643 First Owner 4.0 100.0 Hero   
32644 First Owner 9.0 180.0 TVS   
32645 First Owner 6.0 220.0 Bajaj   
32646 First Owner 16.0 125.0 Hero   
32647 First Owner 13.0 150.0 Bajaj   
  
[32648 rows x 8 columns]

# city=='Delhi'  
delhi = df[df['city']=='Delhi']  
delhi

bike\_name price city kms\_driven \  
1 Royal Enfield Classic 350cc 119900.0 Delhi 11000.0   
2 Triumph Daytona 675R 600000.0 Delhi 110.0   
5 Yamaha FZs 150cc 53499.0 Delhi 25000.0   
6 Honda CB Hornet 160R ABS DLX 85000.0 Delhi 8200.0   
7 Hero Splendor Plus Self Alloy 100cc 45000.0 Delhi 12645.0   
... ... ... ... ...   
32639 Royal Enfield Classic 350cc 95500.0 Delhi 18000.0   
32640 Hero Passion Pro 100cc 32000.0 Delhi 12000.0   
32641 Bajaj Avenger 220cc 41000.0 Delhi 20245.0   
32643 Hero Passion Pro 100cc 39000.0 Delhi 22000.0   
32645 Bajaj Avenger Street 220 60000.0 Delhi 20373.0   
  
 owner age power brand   
1 First Owner 4.0 350.0 Royal Enfield   
2 First Owner 8.0 675.0 Triumph   
5 First Owner 6.0 150.0 Yamaha   
6 First Owner 3.0 160.0 Honda   
7 First Owner 3.0 100.0 Hero   
... ... ... ... ...   
32639 First Owner 8.0 350.0 Royal Enfield   
32640 First Owner 6.0 100.0 Hero   
32641 Second Owner 11.0 220.0 Bajaj   
32643 First Owner 4.0 100.0 Hero   
32645 First Owner 6.0 220.0 Bajaj   
  
[7318 rows x 8 columns]

delhi['owner'].value\_counts()

owner  
First Owner 6516  
Second Owner 787  
Third Owner 13  
Fourth Owner Or More 2  
Name: count, dtype: int64

g = delhi[delhi["owner"]=="Fourth Owner Or More"]

delhi[delhi['owner']=='Fourth Owner Or More']

bike\_name price city kms\_driven \  
2999 Royal Enfield Thunderbird 350cc 95000.0 Delhi 9392.0   
7038 Royal Enfield Bullet Electra 350cc 35000.0 Delhi 60000.0   
  
 owner age power brand   
2999 Fourth Owner Or More 6.0 350.0 Royal Enfield   
7038 Fourth Owner Or More 18.0 350.0 Royal Enfield

delhi\_under60k=delhi[delhi['price']<=60000.0]  
delhi\_under60k

bike\_name price city kms\_driven \  
5 Yamaha FZs 150cc 53499.0 Delhi 25000.0   
7 Hero Splendor Plus Self Alloy 100cc 45000.0 Delhi 12645.0   
13 Bajaj Discover 100M 29499.0 Delhi 20000.0   
14 Bajaj Discover 125M 29900.0 Delhi 20000.0   
19 Bajaj Discover 125M 29900.0 Delhi 20000.0   
... ... ... ... ...   
32637 Bajaj Pulsar 150cc 25000.0 Delhi 32588.0   
32640 Hero Passion Pro 100cc 32000.0 Delhi 12000.0   
32641 Bajaj Avenger 220cc 41000.0 Delhi 20245.0   
32643 Hero Passion Pro 100cc 39000.0 Delhi 22000.0   
32645 Bajaj Avenger Street 220 60000.0 Delhi 20373.0   
  
 owner age power brand   
5 First Owner 6.0 150.0 Yamaha   
7 First Owner 3.0 100.0 Hero   
13 First Owner 8.0 100.0 Bajaj   
14 First Owner 7.0 125.0 Bajaj   
19 First Owner 7.0 125.0 Bajaj   
... ... ... ... ...   
32637 First Owner 9.0 150.0 Bajaj   
32640 First Owner 6.0 100.0 Hero   
32641 Second Owner 11.0 220.0 Bajaj   
32643 First Owner 4.0 100.0 Hero   
32645 First Owner 6.0 220.0 Bajaj   
  
[5221 rows x 8 columns]

delhi\_under60k['brand'].value\_counts()

brand  
Bajaj 2175  
Hero 2087  
Yamaha 738  
Honda 98  
TVS 65  
Suzuki 37  
Royal Enfield 15  
Mahindra 6  
Name: count, dtype: int64

delhi\_Firstowner\_60k=delhi\_under60k[delhi\_under60k['owner']=='First Owner']  
delhi\_Firstowner\_60k

bike\_name price city kms\_driven \  
5 Yamaha FZs 150cc 53499.0 Delhi 25000.0   
7 Hero Splendor Plus Self Alloy 100cc 45000.0 Delhi 12645.0   
13 Bajaj Discover 100M 29499.0 Delhi 20000.0   
14 Bajaj Discover 125M 29900.0 Delhi 20000.0   
19 Bajaj Discover 125M 29900.0 Delhi 20000.0   
... ... ... ... ...   
32622 Yamaha Fazer 150cc 25000.0 Delhi 65000.0   
32637 Bajaj Pulsar 150cc 25000.0 Delhi 32588.0   
32640 Hero Passion Pro 100cc 32000.0 Delhi 12000.0   
32643 Hero Passion Pro 100cc 39000.0 Delhi 22000.0   
32645 Bajaj Avenger Street 220 60000.0 Delhi 20373.0   
  
 owner age power brand   
5 First Owner 6.0 150.0 Yamaha   
7 First Owner 3.0 100.0 Hero   
13 First Owner 8.0 100.0 Bajaj   
14 First Owner 7.0 125.0 Bajaj   
19 First Owner 7.0 125.0 Bajaj   
... ... ... ... ...   
32622 First Owner 12.0 150.0 Yamaha   
32637 First Owner 9.0 150.0 Bajaj   
32640 First Owner 6.0 100.0 Hero   
32643 First Owner 4.0 100.0 Hero   
32645 First Owner 6.0 220.0 Bajaj   
  
[4514 rows x 8 columns]

(delhi\_Firstowner\_60k[["kms\_driven"]].sum())/4514

kms\_driven 32521.431768  
dtype: float64

delhi\_avg\_32k=delhi\_Firstowner\_60k[delhi\_Firstowner\_60k['kms\_driven']<=32521.43176783341]  
delhi\_avg\_32k

bike\_name price city kms\_driven \  
5 Yamaha FZs 150cc 53499.0 Delhi 25000.0   
7 Hero Splendor Plus Self Alloy 100cc 45000.0 Delhi 12645.0   
13 Bajaj Discover 100M 29499.0 Delhi 20000.0   
14 Bajaj Discover 125M 29900.0 Delhi 20000.0   
19 Bajaj Discover 125M 29900.0 Delhi 20000.0   
... ... ... ... ...   
32605 Hero Passion Pro 100cc 39000.0 Delhi 22000.0   
32607 Bajaj Avenger Street 220 60000.0 Delhi 20373.0   
32640 Hero Passion Pro 100cc 32000.0 Delhi 12000.0   
32643 Hero Passion Pro 100cc 39000.0 Delhi 22000.0   
32645 Bajaj Avenger Street 220 60000.0 Delhi 20373.0   
  
 owner age power brand   
5 First Owner 6.0 150.0 Yamaha   
7 First Owner 3.0 100.0 Hero   
13 First Owner 8.0 100.0 Bajaj   
14 First Owner 7.0 125.0 Bajaj   
19 First Owner 7.0 125.0 Bajaj   
... ... ... ... ...   
32605 First Owner 4.0 100.0 Hero   
32607 First Owner 6.0 220.0 Bajaj   
32640 First Owner 6.0 100.0 Hero   
32643 First Owner 4.0 100.0 Hero   
32645 First Owner 6.0 220.0 Bajaj   
  
[2443 rows x 8 columns]

df

bike\_name price city kms\_driven \  
0 TVS Star City Plus Dual Tone 110cc 35000.0 Ahmedabad 17654.0   
1 Royal Enfield Classic 350cc 119900.0 Delhi 11000.0   
2 Triumph Daytona 675R 600000.0 Delhi 110.0   
3 TVS Apache RTR 180cc 65000.0 Bangalore 16329.0   
4 Yamaha FZ S V 2.0 150cc-Ltd. Edition 80000.0 Bangalore 10000.0   
... ... ... ... ...   
32643 Hero Passion Pro 100cc 39000.0 Delhi 22000.0   
32644 TVS Apache RTR 180cc 30000.0 Karnal 6639.0   
32645 Bajaj Avenger Street 220 60000.0 Delhi 20373.0   
32646 Hero Super Splendor 125cc 15600.0 Jaipur 84186.0   
32647 Bajaj Pulsar 150cc 22000.0 Pune 60857.0   
  
 owner age power brand   
0 First Owner 3.0 110.0 TVS   
1 First Owner 4.0 350.0 Royal Enfield   
2 First Owner 8.0 675.0 Triumph   
3 First Owner 4.0 180.0 TVS   
4 First Owner 3.0 150.0 Yamaha   
... ... ... ... ...   
32643 First Owner 4.0 100.0 Hero   
32644 First Owner 9.0 180.0 TVS   
32645 First Owner 6.0 220.0 Bajaj   
32646 First Owner 16.0 125.0 Hero   
32647 First Owner 13.0 150.0 Bajaj   
  
[32648 rows x 8 columns]

df.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 32648 entries, 0 to 32647  
Data columns (total 8 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 bike\_name 32648 non-null object   
 1 price 32648 non-null float64  
 2 city 32648 non-null object   
 3 kms\_driven 32648 non-null float64  
 4 owner 32648 non-null object   
 5 age 32648 non-null float64  
 6 power 32648 non-null float64  
 7 brand 32648 non-null object   
dtypes: float64(4), object(4)  
memory usage: 2.0+ MB

delhi\_avg\_32k['brand'].unique()

array(['Yamaha', 'Hero', 'Bajaj', 'Honda', 'TVS', 'Suzuki',  
 'Royal Enfield', 'Mahindra'], dtype=object)

delhi\_avg\_32k['brand'].value\_counts()

brand  
Hero 1396  
Bajaj 842  
Yamaha 69  
Honda 62  
TVS 43  
Suzuki 25  
Royal Enfield 3  
Mahindra 3  
Name: count, dtype: int64

# brand = 'TVS'  
# age < 3   
# price > 40k  
selected\_bikes = delhi\_avg\_32k[(delhi\_avg\_32k['brand']=='TVS') & (delhi\_avg\_32k['age']<=3) & (delhi\_avg\_32k['price']>40000) & (delhi\_avg\_32k['power']>=180)]  
selected\_bikes

bike\_name price city kms\_driven \  
1515 TVS Apache RTR 180cc 60000.0 Delhi 2400.0   
1526 TVS Apache RTR 180cc 60000.0 Delhi 2400.0   
2226 TVS Apache RTR 200 4V ABS Race Edition 58742.0 Delhi 11068.0   
2251 TVS Apache RTR 200 4V ABS Race Edition 58742.0 Delhi 11068.0   
  
 owner age power brand   
1515 First Owner 3.0 180.0 TVS   
1526 First Owner 3.0 180.0 TVS   
2226 First Owner 3.0 200.0 TVS   
2251 First Owner 3.0 200.0 TVS

selected\_bikes.duplicated().sum()

2

selected\_bikes.drop\_duplicates(inplace=True)

C:\Users\Jai\AppData\Local\Temp\ipykernel\_18232\4105311541.py:1: SettingWithCopyWarning:   
A value is trying to be set on a copy of a slice from a DataFrame  
  
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy  
 selected\_bikes.drop\_duplicates(inplace=True)

selected\_bikes

bike\_name price city kms\_driven \  
1515 TVS Apache RTR 180cc 60000.0 Delhi 2400.0   
2226 TVS Apache RTR 200 4V ABS Race Edition 58742.0 Delhi 11068.0   
  
 owner age power brand   
1515 First Owner 3.0 180.0 TVS   
2226 First Owner 3.0 200.0 TVS

df.duplicated().sum()

25324

df.shape

(32648, 8)

# brand = 'TVS'  
# age < 3   
# price > 40k  
selected\_bikes\_list\_2 = delhi\_avg\_32k[(delhi\_avg\_32k['brand']=='TVS') & (delhi\_avg\_32k['price']>30000) & (delhi\_avg\_32k['power']>=180)]  
selected\_bikes\_list\_2

bike\_name price city kms\_driven \  
1515 TVS Apache RTR 180cc 60000.0 Delhi 2400.0   
1526 TVS Apache RTR 180cc 60000.0 Delhi 2400.0   
2226 TVS Apache RTR 200 4V ABS Race Edition 58742.0 Delhi 11068.0   
2251 TVS Apache RTR 200 4V ABS Race Edition 58742.0 Delhi 11068.0   
3842 TVS Apache RTR 180cc 42000.0 Delhi 30862.0   
5018 TVS Apache RTR 180cc 45000.0 Delhi 18611.0   
7201 TVS Apache RTR 180cc ABS 53000.0 Delhi 29800.0   
8047 TVS Apache RTR 180cc 48000.0 Delhi 18533.0   
  
 owner age power brand   
1515 First Owner 3.0 180.0 TVS   
1526 First Owner 3.0 180.0 TVS   
2226 First Owner 3.0 200.0 TVS   
2251 First Owner 3.0 200.0 TVS   
3842 First Owner 5.0 180.0 TVS   
5018 First Owner 5.0 180.0 TVS   
7201 First Owner 4.0 180.0 TVS   
8047 First Owner 5.0 180.0 TVS

selected\_bikes\_list\_2.to\_csv('filtered\_bike.csv',index=False)

selected\_bikes\_list\_2.to\_excel('filtered\_data.xlsx',index=False)

df

bike\_name price city kms\_driven \  
0 TVS Star City Plus Dual Tone 110cc 35000.0 Ahmedabad 17654.0   
1 Royal Enfield Classic 350cc 119900.0 Delhi 11000.0   
2 Triumph Daytona 675R 600000.0 Delhi 110.0   
3 TVS Apache RTR 180cc 65000.0 Bangalore 16329.0   
4 Yamaha FZ S V 2.0 150cc-Ltd. Edition 80000.0 Bangalore 10000.0   
... ... ... ... ...   
32643 Hero Passion Pro 100cc 39000.0 Delhi 22000.0   
32644 TVS Apache RTR 180cc 30000.0 Karnal 6639.0   
32645 Bajaj Avenger Street 220 60000.0 Delhi 20373.0   
32646 Hero Super Splendor 125cc 15600.0 Jaipur 84186.0   
32647 Bajaj Pulsar 150cc 22000.0 Pune 60857.0   
  
 owner age power brand   
0 First Owner 3.0 110.0 TVS   
1 First Owner 4.0 350.0 Royal Enfield   
2 First Owner 8.0 675.0 Triumph   
3 First Owner 4.0 180.0 TVS   
4 First Owner 3.0 150.0 Yamaha   
... ... ... ... ...   
32643 First Owner 4.0 100.0 Hero   
32644 First Owner 9.0 180.0 TVS   
32645 First Owner 6.0 220.0 Bajaj   
32646 First Owner 16.0 125.0 Hero   
32647 First Owner 13.0 150.0 Bajaj   
  
[32648 rows x 8 columns]

df.drop\_duplicates(inplace=True)

df

bike\_name price city kms\_driven \  
0 TVS Star City Plus Dual Tone 110cc 35000.0 Ahmedabad 17654.0   
1 Royal Enfield Classic 350cc 119900.0 Delhi 11000.0   
2 Triumph Daytona 675R 600000.0 Delhi 110.0   
3 TVS Apache RTR 180cc 65000.0 Bangalore 16329.0   
4 Yamaha FZ S V 2.0 150cc-Ltd. Edition 80000.0 Bangalore 10000.0   
... ... ... ... ...   
9362 Hero Hunk Rear Disc 150cc 25000.0 Delhi 48587.0   
9369 Bajaj Avenger 220cc 35000.0 Bangalore 60000.0   
9370 Harley-Davidson Street 750 ABS 450000.0 Jodhpur 3430.0   
9371 Bajaj Dominar 400 ABS 139000.0 Hyderabad 21300.0   
9372 Bajaj Avenger Street 220 80000.0 Hyderabad 7127.0   
  
 owner age power brand   
0 First Owner 3.0 110.0 TVS   
1 First Owner 4.0 350.0 Royal Enfield   
2 First Owner 8.0 675.0 Triumph   
3 First Owner 4.0 180.0 TVS   
4 First Owner 3.0 150.0 Yamaha   
... ... ... ... ...   
9362 First Owner 8.0 150.0 Hero   
9369 First Owner 9.0 220.0 Bajaj   
9370 First Owner 4.0 750.0 Harley-Davidson   
9371 First Owner 4.0 400.0 Bajaj   
9372 First Owner 5.0 220.0 Bajaj   
  
[7324 rows x 8 columns]

df['brand'].value\_counts()

brand  
Bajaj 2081  
Royal Enfield 1346  
Hero 1142  
Honda 676  
Yamaha 651  
TVS 481  
KTM 375  
Suzuki 203  
Harley-Davidson 91  
Kawasaki 61  
Hyosung 53  
Mahindra 50  
Benelli 46  
Triumph 21  
Ducati 20  
BMW 10  
Jawa 7  
Indian 3  
MV 3  
Rajdoot 1  
LML 1  
Yezdi 1  
Ideal 1  
Name: count, dtype: int64

# brand = ktm   
# owner = first   
# kms driven <15k  
  
KTM\_data= df[(df['brand']=='KTM')&(df['kms\_driven']<=15000)&(df['owner']=='First Owner')]  
KTM\_data

bike\_name price city kms\_driven owner age \  
39 KTM RC 200cc ABS 179000.0 Bangalore 3400.0 First Owner 2.0   
99 KTM Duke 390cc 240000.0 Bangalore 11000.0 First Owner 3.0   
149 KTM Duke 200cc 130000.0 Delhi 9221.0 First Owner 3.0   
179 KTM Duke 200cc 145000.0 Amritsar 4500.0 First Owner 4.0   
199 KTM Duke 200cc 85000.0 Mumbai 15000.0 First Owner 7.0   
... ... ... ... ... ... ...   
8984 KTM Duke 390cc 140000.0 Raigarh(mh) 11100.0 First Owner 6.0   
9011 KTM RC 200cc 150000.0 Delhi 4900.0 First Owner 5.0   
9070 KTM RC 200cc 140000.0 Berhampore 2600.0 First Owner 6.0   
9255 KTM Duke 125cc 135000.0 Bangalore 1050.0 First Owner 3.0   
9348 KTM Duke 390cc 195000.0 Mumbai 4568.0 First Owner 4.0   
  
 power brand   
39 200.0 KTM   
99 390.0 KTM   
149 200.0 KTM   
179 200.0 KTM   
199 200.0 KTM   
... ... ...   
8984 390.0 KTM   
9011 200.0 KTM   
9070 200.0 KTM   
9255 125.0 KTM   
9348 390.0 KTM   
  
[225 rows x 8 columns]

selected\_bikes = delhi\_avg\_32k[(delhi\_avg\_32k['brand']=='TVS') & (delhi\_avg\_32k['age']<=3) & (delhi\_avg\_32k['price']>40000) & (delhi\_avg\_32k['power']>=180)]

# KTM\_data.sort\_values(by='price') # by default ascending order  
KTM\_data.sort\_values(by='price',ascending=False)  
  
  
# 1 90  
# 2 111  
# 3 345  
# 8 50  
  
# ascending price   
# 8 50  
# 1 90  
# 2 111  
# 3 345

bike\_name price city kms\_driven owner age \  
4834 KTM Duke 390cc 263000.0 Delhi 1178.0 First Owner 2.0   
2522 KTM Duke 390cc 260000.0 Vapi 1500.0 First Owner 2.0   
8606 KTM Duke 390cc 250430.0 Jabalpur 4300.0 First Owner 3.0   
8645 KTM Duke 390cc 250000.0 Pune 2730.0 First Owner 2.0   
3508 KTM RC 390cc 250000.0 Bulandshahr 3000.0 First Owner 4.0   
... ... ... ... ... ... ...   
3790 KTM Duke 200cc 75000.0 Chinsurah 13000.0 First Owner 5.0   
2210 KTM Duke 200cc 73000.0 Delhi 7000.0 First Owner 9.0   
7596 KTM Duke 200cc 72500.0 Hyderabad 10000.0 First Owner 8.0   
602 KTM Duke 200cc 72250.0 Pune 6282.0 First Owner 5.0   
1428 KTM Duke 200cc 62240.0 Pune 3706.0 First Owner 7.0   
  
 power brand   
4834 390.0 KTM   
2522 390.0 KTM   
8606 390.0 KTM   
8645 390.0 KTM   
3508 390.0 KTM   
... ... ...   
3790 200.0 KTM   
2210 200.0 KTM   
7596 200.0 KTM   
602 200.0 KTM   
1428 200.0 KTM   
  
[225 rows x 8 columns]

# 1. ktm data -- sort by highest price(top 10),   
# 2 ya toh brand ho ktm ya brand RE (filter(or),RE min price and ktm min price, then tell which bike is available at lowest price) make a function  
# 3 find out avg price of first owner bikes   
# 4 find out all the bikes price sum who has age is 3

df.isnull().sum()

bike\_name 0  
price 0  
city 0  
kms\_driven 0  
owner 0  
age 0  
power 0  
brand 0  
dtype: int64

import numpy as np

data = {  
 'A': [1, 2, 8, 4],  
 'B': ['foo', np.nan, 'bar', np.nan],  
 'C': [5, np.nan, np.nan, 8]  
}  
  
data

{'A': [1, 2, 8, 4], 'B': ['foo', nan, 'bar', nan], 'C': [5, nan, nan, 8]}

df2 = pd.DataFrame(data)  
df2

A B C  
0 1 foo 5.0  
1 2 NaN NaN  
2 8 bar NaN  
3 4 NaN 8.0

df2.isnull().sum()

A 0  
B 2  
C 2  
dtype: int64

df2.isnull().sum().sum()

4

df2

A B C  
0 1 foo 5.0  
1 2 NaN NaN  
2 8 bar NaN  
3 4 NaN 8.0

df2.dropna(axis=0)

A B C  
0 1 foo 5.0

df2.dropna(axis=1)

A  
0 1  
1 2  
2 8  
3 4

df2

A B C  
0 1 foo 5.0  
1 2 NaN NaN  
2 8 bar NaN  
3 4 NaN 8.0

df2.isnull().sum()/df2.shape[0]\*100

A 0.0  
B 50.0  
C 50.0  
dtype: float64

# drop or delete   
# fill

df2

A B C  
0 1 foo 5.0  
1 2 NaN NaN  
2 8 bar NaN  
3 4 NaN 8.0

salary = pd.read\_csv('Salary\_dataset.csv')  
salary.head()

Unnamed: 0 YearsExperience Salary  
0 0 1.2 39344.0  
1 1 1.4 46206.0  
2 2 1.6 37732.0  
3 3 2.1 43526.0  
4 4 2.3 39892.0

salary.shape

(30, 3)

salary.isnull().sum()

Unnamed: 0 0  
YearsExperience 0  
Salary 0  
dtype: int64

df.loc[0:11,['price']] = np.nan

salary.loc[27:,['Salary']]

Salary  
27 112636.0  
28 122392.0  
29 121873.0

salary.isnull().sum()

Unnamed: 0 0  
YearsExperience 0  
Salary 0  
dtype: int64

salary

Unnamed: 0 YearsExperience Salary  
0 0 1.2 39344.0  
1 1 1.4 46206.0  
2 2 1.6 37732.0  
3 3 2.1 43526.0  
4 4 2.3 39892.0  
5 5 3.0 56643.0  
6 6 3.1 60151.0  
7 7 3.3 54446.0  
8 8 3.3 64446.0  
9 9 3.8 57190.0  
10 10 4.0 63219.0  
11 11 4.1 55795.0  
12 12 4.1 56958.0  
13 13 4.2 57082.0  
14 14 4.6 61112.0  
15 15 5.0 67939.0  
16 16 5.2 66030.0  
17 17 5.4 83089.0  
18 18 6.0 81364.0  
19 19 6.1 93941.0  
20 20 6.9 91739.0  
21 21 7.2 98274.0  
22 22 8.0 101303.0  
23 23 8.3 113813.0  
24 24 8.8 109432.0  
25 25 9.1 105583.0  
26 26 9.6 116970.0  
27 27 9.7 112636.0  
28 28 10.4 122392.0  
29 29 10.6 121873.0

# to Fill   
# 1 . constant value fill   
# 2 . mean or median   
#3 . mean or median for each class wise

salary.fillna(25000)

Unnamed: 0 YearsExperience Salary  
0 0 1.2 39344.0  
1 1 1.4 46206.0  
2 2 1.6 37732.0  
3 3 2.1 43526.0  
4 4 2.3 39892.0  
5 5 3.0 56643.0  
6 6 3.1 60151.0  
7 7 3.3 54446.0  
8 8 3.3 64446.0  
9 9 3.8 57190.0  
10 10 4.0 63219.0  
11 11 4.1 55795.0  
12 12 4.1 56958.0  
13 13 4.2 57082.0  
14 14 4.6 61112.0  
15 15 5.0 67939.0  
16 16 5.2 66030.0  
17 17 5.4 83089.0  
18 18 6.0 81364.0  
19 19 6.1 93941.0  
20 20 6.9 91739.0  
21 21 7.2 98274.0  
22 22 8.0 101303.0  
23 23 8.3 113813.0  
24 24 8.8 109432.0  
25 25 9.1 105583.0  
26 26 9.6 116970.0  
27 27 9.7 112636.0  
28 28 10.4 122392.0  
29 29 10.6 121873.0

int(salary['Salary'].mean())

76004

salary.fillna(int(salary['Salary'].mean()))

Unnamed: 0 YearsExperience Salary  
0 0 1.2 39344.0  
1 1 1.4 46206.0  
2 2 1.6 37732.0  
3 3 2.1 43526.0  
4 4 2.3 39892.0  
5 5 3.0 56643.0  
6 6 3.1 60151.0  
7 7 3.3 54446.0  
8 8 3.3 64446.0  
9 9 3.8 57190.0  
10 10 4.0 63219.0  
11 11 4.1 55795.0  
12 12 4.1 56958.0  
13 13 4.2 57082.0  
14 14 4.6 61112.0  
15 15 5.0 67939.0  
16 16 5.2 66030.0  
17 17 5.4 83089.0  
18 18 6.0 81364.0  
19 19 6.1 93941.0  
20 20 6.9 91739.0  
21 21 7.2 98274.0  
22 22 8.0 101303.0  
23 23 8.3 113813.0  
24 24 8.8 109432.0  
25 25 9.1 105583.0  
26 26 9.6 116970.0  
27 27 9.7 112636.0  
28 28 10.4 122392.0  
29 29 10.6 121873.0

# population data  
# sample data

salary.loc[26:,['YearsExperience']]

YearsExperience  
26 9.6  
27 9.7  
28 10.4  
29 10.6

selected\_bikes = delhi\_avg\_32k[(delhi\_avg\_32k['brand']=='TVS') & (delhi\_avg\_32k['age']<=3) & (delhi\_avg\_32k['price']>40000) & (delhi\_avg\_32k['power']>=180)]

filter\_data = salary[(salary['YearsExperience']>=9)]  
filter\_data

Unnamed: 0 YearsExperience Salary  
25 25 9.1 105583.0  
26 26 9.6 116970.0  
27 27 9.7 112636.0  
28 28 10.4 122392.0  
29 29 10.6 121873.0

int(filter\_data['Salary'].mean())

115890

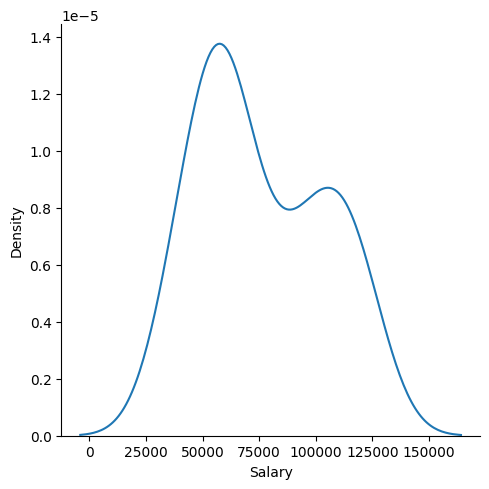
salary.fillna(int(filter\_data['Salary'].mean()))

Unnamed: 0 YearsExperience Salary  
0 0 1.2 39344.0  
1 1 1.4 46206.0  
2 2 1.6 37732.0  
3 3 2.1 43526.0  
4 4 2.3 39892.0  
5 5 3.0 56643.0  
6 6 3.1 60151.0  
7 7 3.3 54446.0  
8 8 3.3 64446.0  
9 9 3.8 57190.0  
10 10 4.0 63219.0  
11 11 4.1 55795.0  
12 12 4.1 56958.0  
13 13 4.2 57082.0  
14 14 4.6 61112.0  
15 15 5.0 67939.0  
16 16 5.2 66030.0  
17 17 5.4 83089.0  
18 18 6.0 81364.0  
19 19 6.1 93941.0  
20 20 6.9 91739.0  
21 21 7.2 98274.0  
22 22 8.0 101303.0  
23 23 8.3 113813.0  
24 24 8.8 109432.0  
25 25 9.1 105583.0  
26 26 9.6 116970.0  
27 27 9.7 112636.0  
28 28 10.4 122392.0  
29 29 10.6 121873.0

import matplotlib.pyplot as plt  
import seaborn as sns

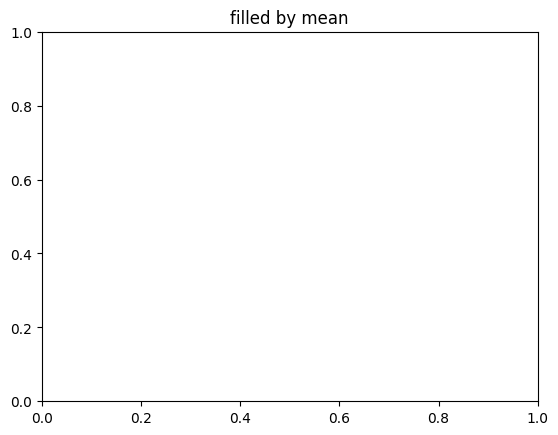
sns.displot(salary['Salary'],kind='kde')

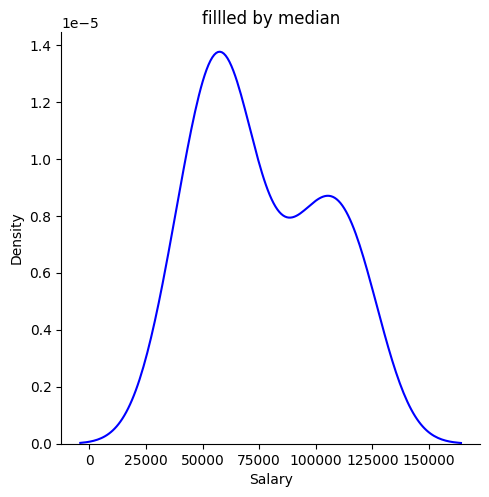
<seaborn.axisgrid.FacetGrid at 0x15d25c4f830>

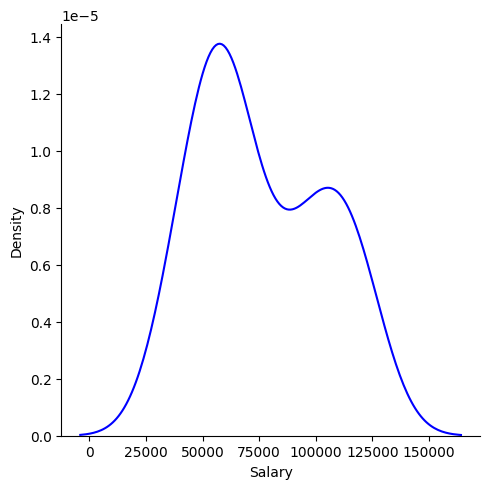


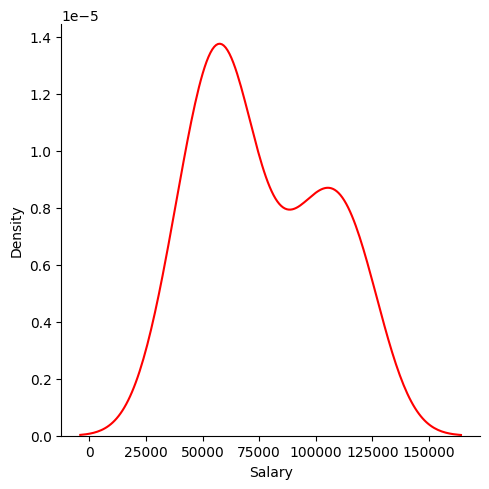
# if your data in normally distributed then used -- > mean to fill missing  
# if not --- > median

mean\_salary = salary.fillna(salary['Salary'].mean())['Salary']  
median\_salary = salary.fillna(salary['Salary'].median())['Salary']  
  
  
  
  
plt.title('filled by mean')  
sns.displot(mean\_salary,kind='kde',color='blue')  
  
plt.title('fillled by median')  
sns.displot(median\_salary,kind='kde',color='blue')  
  
  
  
sns.displot(salary['Salary'],kind='kde',color='red')  
  
plt.show()









salary

Unnamed: 0 YearsExperience Salary  
0 0 1.2 39344.0  
1 1 1.4 46206.0  
2 2 1.6 37732.0  
3 3 2.1 43526.0  
4 4 2.3 39892.0  
5 5 3.0 56643.0  
6 6 3.1 60151.0  
7 7 3.3 54446.0  
8 8 3.3 64446.0  
9 9 3.8 57190.0  
10 10 4.0 63219.0  
11 11 4.1 55795.0  
12 12 4.1 56958.0  
13 13 4.2 57082.0  
14 14 4.6 61112.0  
15 15 5.0 67939.0  
16 16 5.2 66030.0  
17 17 5.4 83089.0  
18 18 6.0 81364.0  
19 19 6.1 93941.0  
20 20 6.9 91739.0  
21 21 7.2 98274.0  
22 22 8.0 101303.0  
23 23 8.3 113813.0  
24 24 8.8 109432.0  
25 25 9.1 105583.0  
26 26 9.6 116970.0  
27 27 9.7 112636.0  
28 28 10.4 122392.0  
29 29 10.6 121873.0

salary.loc[5:10,['YearsExperience']] = np.nan

salary.isnull().sum()

Unnamed: 0 0  
YearsExperience 6  
Salary 0  
dtype: int64

salary.fillna(5)

Unnamed: 0 YearsExperience Salary  
0 0 1.2 39344.0  
1 1 1.4 46206.0  
2 2 1.6 37732.0  
3 3 2.1 43526.0  
4 4 2.3 39892.0  
5 5 5.0 56643.0  
6 6 5.0 60151.0  
7 7 5.0 54446.0  
8 8 5.0 64446.0  
9 9 5.0 57190.0  
10 10 5.0 63219.0  
11 11 4.1 55795.0  
12 12 4.1 56958.0  
13 13 4.2 57082.0  
14 14 4.6 61112.0  
15 15 5.0 67939.0  
16 16 5.2 66030.0  
17 17 5.4 83089.0  
18 18 6.0 81364.0  
19 19 6.1 93941.0  
20 20 6.9 91739.0  
21 21 7.2 98274.0  
22 22 8.0 101303.0  
23 23 8.3 113813.0  
24 24 8.8 109432.0  
25 25 9.1 105583.0  
26 26 9.6 116970.0  
27 27 9.7 112636.0  
28 28 10.4 122392.0  
29 29 10.6 121873.0

salary['Salary'].fillna(1)

0 39344.0  
1 46206.0  
2 37732.0  
3 43526.0  
4 39892.0  
5 56643.0  
6 60151.0  
7 54446.0  
8 64446.0  
9 57190.0  
10 63219.0  
11 55795.0  
12 56958.0  
13 57082.0  
14 61112.0  
15 67939.0  
16 66030.0  
17 83089.0  
18 81364.0  
19 93941.0  
20 91739.0  
21 98274.0  
22 101303.0  
23 113813.0  
24 109432.0  
25 105583.0  
26 116970.0  
27 112636.0  
28 122392.0  
29 121873.0  
Name: Salary, dtype: float64

salary

Unnamed: 0 YearsExperience Salary  
0 0 1.2 39344.0  
1 1 1.4 46206.0  
2 2 1.6 37732.0  
3 3 2.1 43526.0  
4 4 2.3 39892.0  
5 5 NaN 56643.0  
6 6 NaN 60151.0  
7 7 NaN 54446.0  
8 8 NaN 64446.0  
9 9 NaN 57190.0  
10 10 NaN 63219.0  
11 11 4.1 55795.0  
12 12 4.1 56958.0  
13 13 4.2 57082.0  
14 14 4.6 61112.0  
15 15 5.0 67939.0  
16 16 5.2 66030.0  
17 17 5.4 83089.0  
18 18 6.0 81364.0  
19 19 6.1 93941.0  
20 20 6.9 91739.0  
21 21 7.2 98274.0  
22 22 8.0 101303.0  
23 23 8.3 113813.0  
24 24 8.8 109432.0  
25 25 9.1 105583.0  
26 26 9.6 116970.0  
27 27 9.7 112636.0  
28 28 10.4 122392.0  
29 29 10.6 121873.0

salary['Salary'] = salary['Salary']+5000

salary['incre\_salary'] = salary['Salary']+5000

salary

Unnamed: 0 YearsExperience Salary incre\_salary  
0 0 1.2 44344.0 49344.0  
1 1 1.4 51206.0 56206.0  
2 2 1.6 42732.0 47732.0  
3 3 2.1 48526.0 53526.0  
4 4 2.3 44892.0 49892.0  
5 5 NaN 61643.0 66643.0  
6 6 NaN 65151.0 70151.0  
7 7 NaN 59446.0 64446.0  
8 8 NaN 69446.0 74446.0  
9 9 NaN 62190.0 67190.0  
10 10 NaN 68219.0 73219.0  
11 11 4.1 60795.0 65795.0  
12 12 4.1 61958.0 66958.0  
13 13 4.2 62082.0 67082.0  
14 14 4.6 66112.0 71112.0  
15 15 5.0 72939.0 77939.0  
16 16 5.2 71030.0 76030.0  
17 17 5.4 88089.0 93089.0  
18 18 6.0 86364.0 91364.0  
19 19 6.1 98941.0 103941.0  
20 20 6.9 96739.0 101739.0  
21 21 7.2 103274.0 108274.0  
22 22 8.0 106303.0 111303.0  
23 23 8.3 118813.0 123813.0  
24 24 8.8 114432.0 119432.0  
25 25 9.1 110583.0 115583.0  
26 26 9.6 121970.0 126970.0  
27 27 9.7 117636.0 122636.0  
28 28 10.4 127392.0 132392.0  
29 29 10.6 126873.0 131873.0

salary['qanttity'] = 5  
salary['price'] = 100

# "total amount" = quantity \* price

df.head()

bike\_name price city kms\_driven \  
0 TVS Star City Plus Dual Tone 110cc NaN Ahmedabad 17654.0   
1 Royal Enfield Classic 350cc NaN Delhi 11000.0   
2 Triumph Daytona 675R NaN Delhi 110.0   
3 TVS Apache RTR 180cc NaN Bangalore 16329.0   
4 Yamaha FZ S V 2.0 150cc-Ltd. Edition NaN Bangalore 10000.0   
  
 owner age power brand   
0 First Owner 3.0 110.0 TVS   
1 First Owner 4.0 350.0 Royal Enfield   
2 First Owner 8.0 675.0 Triumph   
3 First Owner 4.0 180.0 TVS   
4 First Owner 3.0 150.0 Yamaha

salary['total\_amount'] = salary['qanttity'] \* salary['price']

df.head()

bike\_name price city kms\_driven \  
0 TVS Star City Plus Dual Tone 110cc 35000.0 Ahmedabad 17654.0   
1 Royal Enfield Classic 350cc 119900.0 Delhi 11000.0   
2 Triumph Daytona 675R 600000.0 Delhi 110.0   
3 TVS Apache RTR 180cc 65000.0 Bangalore 16329.0   
4 Yamaha FZ S V 2.0 150cc-Ltd. Edition 80000.0 Bangalore 10000.0   
  
 owner age power brand   
0 First Owner 3.0 110.0 TVS   
1 First Owner 4.0 350.0 Royal Enfield   
2 First Owner 8.0 675.0 Triumph   
3 First Owner 4.0 180.0 TVS   
4 First Owner 3.0 150.0 Yamaha

df["total\_amount"]=df["price"]+df["age"]

df.head()

bike\_name price city kms\_driven \  
0 TVS Star City Plus Dual Tone 110cc 35000.0 Ahmedabad 17654.0   
1 Royal Enfield Classic 350cc 119900.0 Delhi 11000.0   
2 Triumph Daytona 675R 600000.0 Delhi 110.0   
3 TVS Apache RTR 180cc 65000.0 Bangalore 16329.0   
4 Yamaha FZ S V 2.0 150cc-Ltd. Edition 80000.0 Bangalore 10000.0   
  
 owner age power brand total\_amount   
0 First Owner 3.0 110.0 TVS 35003.0   
1 First Owner 4.0 350.0 Royal Enfield 119904.0   
2 First Owner 8.0 675.0 Triumph 600008.0   
3 First Owner 4.0 180.0 TVS 65004.0   
4 First Owner 3.0 150.0 Yamaha 80003.0

salary2 = salary[['YearsExperience','Salary']]  
salary2

YearsExperience Salary  
0 1.2 44344.0  
1 1.4 51206.0  
2 1.6 42732.0  
3 2.1 48526.0  
4 2.3 44892.0  
5 NaN 61643.0  
6 NaN 65151.0  
7 NaN 59446.0  
8 NaN 69446.0  
9 NaN 62190.0  
10 NaN 68219.0  
11 4.1 60795.0  
12 4.1 61958.0  
13 4.2 62082.0  
14 4.6 66112.0  
15 5.0 72939.0  
16 5.2 71030.0  
17 5.4 88089.0  
18 6.0 86364.0  
19 6.1 98941.0  
20 6.9 96739.0  
21 7.2 103274.0  
22 8.0 106303.0  
23 8.3 118813.0  
24 8.8 114432.0  
25 9.1 110583.0  
26 9.6 121970.0  
27 9.7 117636.0  
28 10.4 127392.0  
29 10.6 126873.0

salary

Unnamed: 0 YearsExperience Salary incre\_salary qanttity price \  
0 0 1.2 44344.0 49344.0 5 100   
1 1 1.4 51206.0 56206.0 5 100   
2 2 1.6 42732.0 47732.0 5 100   
3 3 2.1 48526.0 53526.0 5 100   
4 4 2.3 44892.0 49892.0 5 100   
5 5 NaN 61643.0 66643.0 5 100   
6 6 NaN 65151.0 70151.0 5 100   
7 7 NaN 59446.0 64446.0 5 100   
8 8 NaN 69446.0 74446.0 5 100   
9 9 NaN 62190.0 67190.0 5 100   
10 10 NaN 68219.0 73219.0 5 100   
11 11 4.1 60795.0 65795.0 5 100   
12 12 4.1 61958.0 66958.0 5 100   
13 13 4.2 62082.0 67082.0 5 100   
14 14 4.6 66112.0 71112.0 5 100   
15 15 5.0 72939.0 77939.0 5 100   
16 16 5.2 71030.0 76030.0 5 100   
17 17 5.4 88089.0 93089.0 5 100   
18 18 6.0 86364.0 91364.0 5 100   
19 19 6.1 98941.0 103941.0 5 100   
20 20 6.9 96739.0 101739.0 5 100   
21 21 7.2 103274.0 108274.0 5 100   
22 22 8.0 106303.0 111303.0 5 100   
23 23 8.3 118813.0 123813.0 5 100   
24 24 8.8 114432.0 119432.0 5 100   
25 25 9.1 110583.0 115583.0 5 100   
26 26 9.6 121970.0 126970.0 5 100   
27 27 9.7 117636.0 122636.0 5 100   
28 28 10.4 127392.0 132392.0 5 100   
29 29 10.6 126873.0 131873.0 5 100   
  
 total\_amount   
0 500   
1 500   
2 500   
3 500   
4 500   
5 500   
6 500   
7 500   
8 500   
9 500   
10 500   
11 500   
12 500   
13 500   
14 500   
15 500   
16 500   
17 500   
18 500   
19 500   
20 500   
21 500   
22 500   
23 500   
24 500   
25 500   
26 500   
27 500   
28 500   
29 500

df.head()

bike\_name price city kms\_driven \  
0 TVS Star City Plus Dual Tone 110cc NaN Ahmedabad 17654.0   
1 Royal Enfield Classic 350cc NaN Delhi 11000.0   
2 Triumph Daytona 675R NaN Delhi 110.0   
3 TVS Apache RTR 180cc NaN Bangalore 16329.0   
4 Yamaha FZ S V 2.0 150cc-Ltd. Edition NaN Bangalore 10000.0   
  
 owner age power brand   
0 First Owner 3.0 110.0 TVS   
1 First Owner 4.0 350.0 Royal Enfield   
2 First Owner 8.0 675.0 Triumph   
3 First Owner 4.0 180.0 TVS   
4 First Owner 3.0 150.0 Yamaha

salary.drop('constant',axis=1)

---------------------------------------------------------------------------  
KeyError Traceback (most recent call last)  
Cell In[278], line 1  
----> 1 salary.drop('constant',axis=1)  
  
File c:\Users\Jai\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\frame.py:5568, in DataFrame.drop(self, labels, axis, index, columns, level, inplace, errors)  
 5420 def drop(  
 5421 self,  
 5422 labels: IndexLabel | None = None,  
 (...)  
 5429 errors: IgnoreRaise = "raise",  
 5430 ) -> DataFrame | None:  
 5431 """  
 5432 Drop specified labels from rows or columns.  
 5433   
 (...)  
 5566 weight 1.0 0.8  
 5567 """  
-> 5568 return super().drop(  
 5569 labels=labels,  
 5570 axis=axis,  
 5571 index=index,  
 5572 columns=columns,  
 5573 level=level,  
 5574 inplace=inplace,  
 5575 errors=errors,  
 5576 )  
  
File c:\Users\Jai\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\generic.py:4785, in NDFrame.drop(self, labels, axis, index, columns, level, inplace, errors)  
 4783 for axis, labels in axes.items():  
 4784 if labels is not None:  
-> 4785 obj = obj.\_drop\_axis(labels, axis, level=level, errors=errors)  
 4787 if inplace:  
 4788 self.\_update\_inplace(obj)  
  
File c:\Users\Jai\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\generic.py:4827, in NDFrame.\_drop\_axis(self, labels, axis, level, errors, only\_slice)  
 4825 new\_axis = axis.drop(labels, level=level, errors=errors)  
 4826 else:  
-> 4827 new\_axis = axis.drop(labels, errors=errors)  
 4828 indexer = axis.get\_indexer(new\_axis)  
 4830 # Case for non-unique axis  
 4831 else:  
  
File c:\Users\Jai\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\indexes\base.py:7070, in Index.drop(self, labels, errors)  
 7068 if mask.any():  
 7069 if errors != "ignore":  
-> 7070 raise KeyError(f"{labels[mask].tolist()} not found in axis")  
 7071 indexer = indexer[~mask]  
 7072 return self.delete(indexer)  
  
KeyError: "['constant'] not found in axis"

salary.drop(['constant','qanttity','price','total\_amount'],axis=1,inplace=True)

---------------------------------------------------------------------------  
KeyError Traceback (most recent call last)  
Cell In[140], line 1  
----> 1 salary.drop(['constant','qanttity','price','total\_amount'],axis=1,inplace=True)  
  
File c:\Users\Jai\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\frame.py:5568, in DataFrame.drop(self, labels, axis, index, columns, level, inplace, errors)  
 5420 def drop(  
 5421 self,  
 5422 labels: IndexLabel | None = None,  
 (...)  
 5429 errors: IgnoreRaise = "raise",  
 5430 ) -> DataFrame | None:  
 5431 """  
 5432 Drop specified labels from rows or columns.  
 5433   
 (...)  
 5566 weight 1.0 0.8  
 5567 """  
-> 5568 return super().drop(  
 5569 labels=labels,  
 5570 axis=axis,  
 5571 index=index,  
 5572 columns=columns,  
 5573 level=level,  
 5574 inplace=inplace,  
 5575 errors=errors,  
 5576 )  
  
File c:\Users\Jai\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\generic.py:4785, in NDFrame.drop(self, labels, axis, index, columns, level, inplace, errors)  
 4783 for axis, labels in axes.items():  
 4784 if labels is not None:  
-> 4785 obj = obj.\_drop\_axis(labels, axis, level=level, errors=errors)  
 4787 if inplace:  
 4788 self.\_update\_inplace(obj)  
  
File c:\Users\Jai\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\generic.py:4827, in NDFrame.\_drop\_axis(self, labels, axis, level, errors, only\_slice)  
 4825 new\_axis = axis.drop(labels, level=level, errors=errors)  
 4826 else:  
-> 4827 new\_axis = axis.drop(labels, errors=errors)  
 4828 indexer = axis.get\_indexer(new\_axis)  
 4830 # Case for non-unique axis  
 4831 else:  
  
File c:\Users\Jai\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\indexes\base.py:7070, in Index.drop(self, labels, errors)  
 7068 if mask.any():  
 7069 if errors != "ignore":  
-> 7070 raise KeyError(f"{labels[mask].tolist()} not found in axis")  
 7071 indexer = indexer[~mask]  
 7072 return self.delete(indexer)  
  
KeyError: "['constant', 'total\_amount'] not found in axis"

salary.dropna(inplace=True)

salary.drop('Unnamed: 0',axis=1,inplace=True)

salary.columns

Index(['YearsExperience', 'Salary', 'incre\_salary'], dtype='object')

salary.columns = ['YOE', 'salary', 'incremented']

---------------------------------------------------------------------------  
ValueError Traceback (most recent call last)  
Cell In[139], line 1  
----> 1 salary.columns = ['YOE', 'salary', 'incremented']  
  
File c:\Users\Jai\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\generic.py:6310, in NDFrame.\_\_setattr\_\_(self, name, value)  
 6308 try:  
 6309 object.\_\_getattribute\_\_(self, name)  
-> 6310 return object.\_\_setattr\_\_(self, name, value)  
 6311 except AttributeError:  
 6312 pass  
  
File properties.pyx:69, in pandas.\_libs.properties.AxisProperty.\_\_set\_\_()  
  
File c:\Users\Jai\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\generic.py:813, in NDFrame.\_set\_axis(self, axis, labels)  
 808 """  
 809 This is called from the cython code when we set the `index` attribute  
 810 directly, e.g. `series.index = [1, 2, 3]`.  
 811 """  
 812 labels = ensure\_index(labels)  
--> 813 self.\_mgr.set\_axis(axis, labels)  
 814 self.\_clear\_item\_cache()  
  
File c:\Users\Jai\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\internals\managers.py:238, in BaseBlockManager.set\_axis(self, axis, new\_labels)  
 236 def set\_axis(self, axis: AxisInt, new\_labels: Index) -> None:  
 237 # Caller is responsible for ensuring we have an Index object.  
--> 238 self.\_validate\_set\_axis(axis, new\_labels)  
 239 self.axes[axis] = new\_labels  
  
File c:\Users\Jai\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\internals\base.py:98, in DataManager.\_validate\_set\_axis(self, axis, new\_labels)  
 95 pass  
 97 elif new\_len != old\_len:  
---> 98 raise ValueError(  
 99 f"Length mismatch: Expected axis has {old\_len} elements, new "  
 100 f"values have {new\_len} elements"  
 101 )  
  
ValueError: Length mismatch: Expected axis has 6 elements, new values have 3 elements

salary

YOE salary incremented  
0 1.2 44344.0 49344.0  
1 1.4 51206.0 56206.0  
2 1.6 42732.0 47732.0  
3 2.1 48526.0 53526.0  
4 2.3 44892.0 49892.0  
11 4.1 60795.0 65795.0  
12 4.1 61958.0 66958.0  
13 4.2 62082.0 67082.0  
14 4.6 66112.0 71112.0  
15 5.0 72939.0 77939.0  
16 5.2 71030.0 76030.0  
17 5.4 88089.0 93089.0  
18 6.0 86364.0 91364.0  
19 6.1 98941.0 103941.0  
20 6.9 96739.0 101739.0  
21 7.2 103274.0 108274.0  
22 8.0 106303.0 111303.0  
23 8.3 118813.0 123813.0  
24 8.8 114432.0 119432.0  
25 9.1 110583.0 115583.0  
26 9.6 121970.0 126970.0

salary.rename({'YOE':'yearofexp'},inplace=True,axis=1)

salary['salary'] = salary['salary'].astype('int')

salary

Unnamed: 0 YearsExperience Salary incre\_salary qanttity price \  
0 0 1.2 44344.0 49344.0 5 100   
1 1 1.4 51206.0 56206.0 5 100   
2 2 1.6 42732.0 47732.0 5 100   
3 3 2.1 48526.0 53526.0 5 100   
4 4 2.3 44892.0 49892.0 5 100   
5 5 NaN 61643.0 66643.0 5 100   
6 6 NaN 65151.0 70151.0 5 100   
7 7 NaN 59446.0 64446.0 5 100   
8 8 NaN 69446.0 74446.0 5 100   
9 9 NaN 62190.0 67190.0 5 100   
10 10 NaN 68219.0 73219.0 5 100   
11 11 4.1 60795.0 65795.0 5 100   
12 12 4.1 61958.0 66958.0 5 100   
13 13 4.2 62082.0 67082.0 5 100   
14 14 4.6 66112.0 71112.0 5 100   
15 15 5.0 72939.0 77939.0 5 100   
16 16 5.2 71030.0 76030.0 5 100   
17 17 5.4 88089.0 93089.0 5 100   
18 18 6.0 86364.0 91364.0 5 100   
19 19 6.1 98941.0 103941.0 5 100   
20 20 6.9 96739.0 101739.0 5 100   
21 21 7.2 103274.0 108274.0 5 100   
22 22 8.0 106303.0 111303.0 5 100   
23 23 8.3 118813.0 123813.0 5 100   
24 24 8.8 114432.0 119432.0 5 100   
25 25 9.1 110583.0 115583.0 5 100   
26 26 9.6 121970.0 126970.0 5 100   
27 27 9.7 117636.0 122636.0 5 100   
28 28 10.4 127392.0 132392.0 5 100   
29 29 10.6 126873.0 131873.0 5 100   
  
 total\_amount   
0 500   
1 500   
2 500   
3 500   
4 500   
5 500   
6 500   
7 500   
8 500   
9 500   
10 500   
11 500   
12 500   
13 500   
14 500   
15 500   
16 500   
17 500   
18 500   
19 500   
20 500   
21 500   
22 500   
23 500   
24 500   
25 500   
26 500   
27 500   
28 500   
29 500

import pandas as pd

df = pd.read\_csv('laptopPrice.csv')  
df.head()

brand processor\_brand processor\_name processor\_gnrtn ram\_gb ram\_type \  
0 ASUS Intel Core i3 10th 4 GB DDR4   
1 Lenovo Intel Core i3 10th 4 GB DDR4   
2 Lenovo Intel Core i3 10th 4 GB DDR4   
3 ASUS Intel Core i5 10th 8 GB DDR4   
4 ASUS Intel Celeron Dual Not Available 4 GB DDR4   
  
 ssd hdd os os\_bit graphic\_card\_gb weight warranty \  
0 0 GB 1024 GB Windows 64-bit 0 GB Casual No warranty   
1 0 GB 1024 GB Windows 64-bit 0 GB Casual No warranty   
2 0 GB 1024 GB Windows 64-bit 0 GB Casual No warranty   
3 512 GB 0 GB Windows 32-bit 2 GB Casual No warranty   
4 0 GB 512 GB Windows 64-bit 0 GB Casual No warranty   
  
 Touchscreen msoffice Price rating Number of Ratings Number of Reviews   
0 No No 34649 2 stars 3 0   
1 No No 38999 3 stars 65 5   
2 No No 39999 3 stars 8 1   
3 No No 69990 3 stars 0 0   
4 No No 26990 3 stars 0 0

bike = pd.read\_csv('Used\_Bikes.csv')  
bike.head()

bike\_name price city kms\_driven \  
0 TVS Star City Plus Dual Tone 110cc 35000.0 Ahmedabad 17654.0   
1 Royal Enfield Classic 350cc 119900.0 Delhi 11000.0   
2 Triumph Daytona 675R 600000.0 Delhi 110.0   
3 TVS Apache RTR 180cc 65000.0 Bangalore 16329.0   
4 Yamaha FZ S V 2.0 150cc-Ltd. Edition 80000.0 Bangalore 10000.0   
  
 owner age power brand   
0 First Owner 3.0 110.0 TVS   
1 First Owner 4.0 350.0 Royal Enfield   
2 First Owner 8.0 675.0 Triumph   
3 First Owner 4.0 180.0 TVS   
4 First Owner 3.0 150.0 Yamaha

owner\_group = bike.groupby('owner')  
owner\_group

<pandas.core.groupby.generic.DataFrameGroupBy object at 0x0000015D2774A180>

bike['owner'].value\_counts()

owner  
First Owner 29964  
Second Owner 2564  
Third Owner 108  
Fourth Owner Or More 12  
Name: count, dtype: int64

owner\_group.get\_group('First Owner')

bike\_name price city kms\_driven \  
0 TVS Star City Plus Dual Tone 110cc 35000.0 Ahmedabad 17654.0   
1 Royal Enfield Classic 350cc 119900.0 Delhi 11000.0   
2 Triumph Daytona 675R 600000.0 Delhi 110.0   
3 TVS Apache RTR 180cc 65000.0 Bangalore 16329.0   
4 Yamaha FZ S V 2.0 150cc-Ltd. Edition 80000.0 Bangalore 10000.0   
... ... ... ... ...   
32643 Hero Passion Pro 100cc 39000.0 Delhi 22000.0   
32644 TVS Apache RTR 180cc 30000.0 Karnal 6639.0   
32645 Bajaj Avenger Street 220 60000.0 Delhi 20373.0   
32646 Hero Super Splendor 125cc 15600.0 Jaipur 84186.0   
32647 Bajaj Pulsar 150cc 22000.0 Pune 60857.0   
  
 owner age power brand   
0 First Owner 3.0 110.0 TVS   
1 First Owner 4.0 350.0 Royal Enfield   
2 First Owner 8.0 675.0 Triumph   
3 First Owner 4.0 180.0 TVS   
4 First Owner 3.0 150.0 Yamaha   
... ... ... ... ...   
32643 First Owner 4.0 100.0 Hero   
32644 First Owner 9.0 180.0 TVS   
32645 First Owner 6.0 220.0 Bajaj   
32646 First Owner 16.0 125.0 Hero   
32647 First Owner 13.0 150.0 Bajaj   
  
[29964 rows x 8 columns]

owner\_group.get\_group('Third Owner').head()

bike\_name price city kms\_driven \  
103 Bajaj Pulsar 150cc 11100.0 Pune 12000.0   
225 KTM RC 200cc 113000.0 Kottayam 28000.0   
284 Triumph Street Triple ABS 675cc 599999.0 Mumbai 7800.0   
364 Kawasaki Ninja 250cc 125000.0 Coimbatore 21000.0   
474 Mahindra Centuro Rockstar 110cc 18999.0 Pune 38669.0   
  
 owner age power brand   
103 Third Owner 12.0 150.0 Bajaj   
225 Third Owner 6.0 200.0 KTM   
284 Third Owner 5.0 675.0 Triumph   
364 Third Owner 11.0 250.0 Kawasaki   
474 Third Owner 6.0 110.0 Mahindra

owner\_group['price'].min()

owner  
First Owner 4400.0  
Fourth Owner Or More 11000.0  
Second Owner 7500.0  
Third Owner 7000.0  
Name: price, dtype: float64

bike[bike['owner']=='Second Owner']['price'].min()

7500.0

owner\_group['age'].min()

owner  
First Owner 1.0  
Fourth Owner Or More 4.0  
Second Owner 2.0  
Third Owner 4.0  
Name: age, dtype: float64

owner\_group['age'].max()

owner  
First Owner 43.0  
Fourth Owner Or More 37.0  
Second Owner 63.0  
Third Owner 43.0  
Name: age, dtype: float64

age = owner\_group['age'].agg(mini\_age='min',max\_age='max',mean\_age='mean')  
age

mini\_age max\_age mean\_age  
owner   
First Owner 1.0 43.0 7.684855  
Fourth Owner Or More 4.0 37.0 15.250000  
Second Owner 2.0 63.0 12.129095  
Third Owner 4.0 43.0 11.175926

age.reset\_index()

owner mini\_age max\_age mean\_age  
0 First Owner 1.0 43.0 7.684855  
1 Fourth Owner Or More 4.0 37.0 15.250000  
2 Second Owner 2.0 63.0 12.129095  
3 Third Owner 4.0 43.0 11.175926

bike  
  
postal code   
  
  
owner , degree  
1  
2  
3  
2

bike\_name price city kms\_driven \  
0 TVS Star City Plus Dual Tone 110cc 35000.0 Ahmedabad 17654.0   
1 Royal Enfield Classic 350cc 119900.0 Delhi 11000.0   
2 Triumph Daytona 675R 600000.0 Delhi 110.0   
3 TVS Apache RTR 180cc 65000.0 Bangalore 16329.0   
4 Yamaha FZ S V 2.0 150cc-Ltd. Edition 80000.0 Bangalore 10000.0   
... ... ... ... ...   
32643 Hero Passion Pro 100cc 39000.0 Delhi 22000.0   
32644 TVS Apache RTR 180cc 30000.0 Karnal 6639.0   
32645 Bajaj Avenger Street 220 60000.0 Delhi 20373.0   
32646 Hero Super Splendor 125cc 15600.0 Jaipur 84186.0   
32647 Bajaj Pulsar 150cc 22000.0 Pune 60857.0   
  
 owner age power brand   
0 First Owner 3.0 110.0 TVS   
1 First Owner 4.0 350.0 Royal Enfield   
2 First Owner 8.0 675.0 Triumph   
3 First Owner 4.0 180.0 TVS   
4 First Owner 3.0 150.0 Yamaha   
... ... ... ... ...   
32643 First Owner 4.0 100.0 Hero   
32644 First Owner 9.0 180.0 TVS   
32645 First Owner 6.0 220.0 Bajaj   
32646 First Owner 16.0 125.0 Hero   
32647 First Owner 13.0 150.0 Bajaj   
  
[32648 rows x 8 columns]

brand\_group = df.groupby('brand')

brand\_group.get\_group('ASUS')

brand processor\_brand processor\_name processor\_gnrtn ram\_gb ram\_type \  
0 ASUS Intel Core i3 10th 4 GB DDR4   
3 ASUS Intel Core i5 10th 8 GB DDR4   
4 ASUS Intel Celeron Dual Not Available 4 GB DDR4   
5 ASUS Intel Celeron Dual Not Available 4 GB DDR4   
6 ASUS Intel Celeron Dual Not Available 4 GB DDR4   
.. ... ... ... ... ... ...   
817 ASUS AMD Ryzen 9 Not Available 4 GB LPDDR4X   
818 ASUS AMD Ryzen 9 Not Available 4 GB DDR4   
819 ASUS AMD Ryzen 9 Not Available 4 GB DDR4   
820 ASUS AMD Ryzen 9 Not Available 4 GB DDR4   
821 ASUS AMD Ryzen 9 Not Available 4 GB DDR4   
  
 ssd hdd os os\_bit graphic\_card\_gb weight warranty \  
0 0 GB 1024 GB Windows 64-bit 0 GB Casual No warranty   
3 512 GB 0 GB Windows 32-bit 2 GB Casual No warranty   
4 0 GB 512 GB Windows 64-bit 0 GB Casual No warranty   
5 0 GB 512 GB Windows 64-bit 0 GB Casual No warranty   
6 0 GB 512 GB Windows 32-bit 0 GB Casual No warranty   
.. ... ... ... ... ... ... ...   
817 1024 GB 0 GB Windows 64-bit 4 GB Casual 1 year   
818 1024 GB 0 GB Windows 64-bit 0 GB Casual 1 year   
819 1024 GB 0 GB Windows 64-bit 0 GB Casual 1 year   
820 1024 GB 0 GB Windows 64-bit 4 GB Casual 1 year   
821 1024 GB 0 GB Windows 64-bit 4 GB Casual 1 year   
  
 Touchscreen msoffice Price rating Number of Ratings \  
0 No No 34649 2 stars 3   
3 No No 69990 3 stars 0   
4 No No 26990 3 stars 0   
5 No No 22990 3 stars 0   
6 No No 21990 3 stars 31   
.. ... ... ... ... ...   
817 Yes Yes 234990 3 stars 0   
818 No No 135990 3 stars 0   
819 No No 144990 3 stars 0   
820 No No 149990 3 stars 0   
821 No No 142990 3 stars 0   
  
 Number of Reviews   
0 0   
3 0   
4 0   
5 0   
6 3   
.. ...   
817 0   
818 0   
819 0   
820 0   
821 0   
  
[249 rows x 19 columns]

pd.DataFrame(brand\_group['ssd'].value\_counts())

count  
brand ssd   
APPLE 512 GB 13  
 1024 GB 8  
 256 GB 7  
ASUS 512 GB 99  
 1024 GB 59  
 256 GB 49  
 0 GB 39  
 128 GB 1  
 2048 GB 1  
 3072 GB 1  
Avita 256 GB 6  
 512 GB 6  
 0 GB 1  
 1024 GB 1  
DELL 512 GB 65  
 256 GB 57  
 0 GB 25  
 1024 GB 4  
HP 512 GB 83  
 0 GB 20  
 256 GB 17  
 1024 GB 14  
 128 GB 2  
Lenovo 512 GB 85  
 256 GB 31  
 0 GB 16  
 1024 GB 12  
MSI 512 GB 29  
 1024 GB 9  
 0 GB 8  
 256 GB 4  
acer 512 GB 18  
 0 GB 15  
 256 GB 13  
 1024 GB 4  
 2048 GB 1

df

brand processor\_brand processor\_name processor\_gnrtn ram\_gb ram\_type \  
0 ASUS Intel Core i3 10th 4 GB DDR4   
1 Lenovo Intel Core i3 10th 4 GB DDR4   
2 Lenovo Intel Core i3 10th 4 GB DDR4   
3 ASUS Intel Core i5 10th 8 GB DDR4   
4 ASUS Intel Celeron Dual Not Available 4 GB DDR4   
.. ... ... ... ... ... ...   
818 ASUS AMD Ryzen 9 Not Available 4 GB DDR4   
819 ASUS AMD Ryzen 9 Not Available 4 GB DDR4   
820 ASUS AMD Ryzen 9 Not Available 4 GB DDR4   
821 ASUS AMD Ryzen 9 Not Available 4 GB DDR4   
822 Lenovo AMD Ryzen 5 10th 8 GB DDR4   
  
 ssd hdd os os\_bit graphic\_card\_gb weight \  
0 0 GB 1024 GB Windows 64-bit 0 GB Casual   
1 0 GB 1024 GB Windows 64-bit 0 GB Casual   
2 0 GB 1024 GB Windows 64-bit 0 GB Casual   
3 512 GB 0 GB Windows 32-bit 2 GB Casual   
4 0 GB 512 GB Windows 64-bit 0 GB Casual   
.. ... ... ... ... ... ...   
818 1024 GB 0 GB Windows 64-bit 0 GB Casual   
819 1024 GB 0 GB Windows 64-bit 0 GB Casual   
820 1024 GB 0 GB Windows 64-bit 4 GB Casual   
821 1024 GB 0 GB Windows 64-bit 4 GB Casual   
822 512 GB 0 GB DOS 64-bit 0 GB ThinNlight   
  
 warranty Touchscreen msoffice Price rating Number of Ratings \  
0 No warranty No No 34649 2 stars 3   
1 No warranty No No 38999 3 stars 65   
2 No warranty No No 39999 3 stars 8   
3 No warranty No No 69990 3 stars 0   
4 No warranty No No 26990 3 stars 0   
.. ... ... ... ... ... ...   
818 1 year No No 135990 3 stars 0   
819 1 year No No 144990 3 stars 0   
820 1 year No No 149990 3 stars 0   
821 1 year No No 142990 3 stars 0   
822 No warranty No No 57490 4 stars 18   
  
 Number of Reviews   
0 0   
1 5   
2 1   
3 0   
4 0   
.. ...   
818 0   
819 0   
820 0   
821 0   
822 4   
  
[823 rows x 19 columns]

pd.DataFrame(brand\_group[['ssd','hdd','Touchscreen']].value\_counts())

count  
brand ssd hdd Touchscreen   
APPLE 512 GB 0 GB No 13  
 1024 GB 0 GB No 8  
 256 GB 0 GB No 7  
ASUS 512 GB 0 GB No 85  
 1024 GB 0 GB No 45  
... ...  
acer 256 GB 0 GB No 4  
 512 GB 1024 GB No 2  
 2048 GB 0 GB No 1  
 256 GB 0 GB Yes 1  
 512 GB 0 GB Yes 1  
  
[67 rows x 1 columns]

df

pd.DataFrame(brand\_group[['Price','Number of Ratings']].mean())

Price Number of Ratings  
brand   
APPLE 151707.857143 1207.535714  
ASUS 78937.602410 363.244980  
Avita 65157.428571 98.428571  
DELL 60934.662252 124.761589  
HP 73640.272059 447.198529  
Lenovo 72920.208333 160.090278  
MSI 98713.020000 253.380000  
acer 72420.039216 362.274510

dell = df[(df['brand']=='DELL') & (df['ram\_gb']=='4 GB')]  
dell

brand processor\_brand processor\_name processor\_gnrtn ram\_gb ram\_type \  
40 DELL Intel Core i3 10th 4 GB DDR4   
44 DELL Intel Core i3 10th 4 GB DDR4   
59 DELL Intel Core i3 11th 4 GB DDR4   
63 DELL Intel Core i3 10th 4 GB DDR4   
65 DELL Intel Core i3 11th 4 GB DDR4   
76 DELL Intel Core i3 11th 4 GB DDR4   
77 DELL Intel Core i3 11th 4 GB DDR4   
79 DELL Intel Core i3 10th 4 GB DDR4   
126 DELL Intel Core i3 10th 4 GB DDR4   
141 DELL Intel Core i3 10th 4 GB DDR4   
142 DELL Intel Core i3 10th 4 GB DDR4   
146 DELL Intel Core i3 11th 4 GB DDR4   
152 DELL Intel Core i3 10th 4 GB DDR4   
155 DELL Intel Core i3 10th 4 GB DDR4   
175 DELL Intel Core i3 11th 4 GB DDR4   
178 DELL Intel Core i3 11th 4 GB DDR4   
179 DELL Intel Core i3 10th 4 GB DDR4   
181 DELL Intel Core i3 11th 4 GB DDR4   
182 DELL Intel Core i3 11th 4 GB DDR4   
183 DELL Intel Core i3 10th 4 GB DDR4   
184 DELL Intel Core i3 10th 4 GB DDR4   
185 DELL Intel Core i3 11th 4 GB DDR4   
186 DELL Intel Core i3 11th 4 GB DDR4   
455 DELL Intel Core i5 11th 4 GB DDR4   
629 DELL Intel Pentium Quad Not Available 4 GB DDR4   
631 DELL Intel Pentium Quad Not Available 4 GB DDR4   
650 DELL AMD Ryzen 3 Not Available 4 GB DDR4   
655 DELL AMD Ryzen 3 Not Available 4 GB DDR4   
656 DELL AMD Ryzen 3 Not Available 4 GB DDR4   
657 DELL AMD Ryzen 3 Not Available 4 GB DDR4   
667 DELL AMD Ryzen 5 Not Available 4 GB DDR4   
698 DELL AMD Ryzen 5 Not Available 4 GB DDR4   
699 DELL AMD Ryzen 5 Not Available 4 GB DDR4   
710 DELL AMD Ryzen 5 Not Available 4 GB DDR4   
721 DELL AMD Ryzen 5 Not Available 4 GB DDR4   
726 DELL AMD Ryzen 5 Not Available 4 GB DDR4   
736 DELL AMD Ryzen 5 Not Available 4 GB DDR4   
764 DELL AMD Ryzen 7 Not Available 4 GB DDR4   
769 DELL AMD Ryzen 7 Not Available 4 GB DDR4   
  
 ssd hdd os os\_bit graphic\_card\_gb weight \  
40 256 GB 0 GB Windows 64-bit 0 GB ThinNlight   
44 256 GB 0 GB Windows 64-bit 0 GB Casual   
59 256 GB 0 GB Windows 64-bit 0 GB ThinNlight   
63 256 GB 0 GB Windows 64-bit 0 GB ThinNlight   
65 256 GB 0 GB Windows 64-bit 0 GB Casual   
76 256 GB 0 GB Windows 32-bit 0 GB Casual   
77 256 GB 0 GB Windows 64-bit 0 GB Casual   
79 256 GB 0 GB Windows 64-bit 0 GB ThinNlight   
126 512 GB 0 GB Windows 64-bit 0 GB Casual   
141 0 GB 1024 GB Windows 64-bit 0 GB ThinNlight   
142 0 GB 1024 GB Windows 32-bit 0 GB Casual   
146 0 GB 1024 GB Windows 32-bit 0 GB Casual   
152 0 GB 1024 GB Windows 64-bit 0 GB Casual   
155 0 GB 1024 GB Windows 64-bit 0 GB ThinNlight   
175 256 GB 1024 GB Windows 64-bit 0 GB ThinNlight   
178 256 GB 1024 GB Windows 64-bit 0 GB ThinNlight   
179 256 GB 1024 GB Windows 64-bit 0 GB Casual   
181 256 GB 1024 GB Windows 64-bit 0 GB ThinNlight   
182 256 GB 1024 GB Windows 64-bit 0 GB ThinNlight   
183 256 GB 1024 GB Windows 64-bit 0 GB ThinNlight   
184 256 GB 1024 GB Windows 64-bit 0 GB ThinNlight   
185 256 GB 1024 GB Windows 64-bit 0 GB Casual   
186 256 GB 1024 GB Windows 64-bit 0 GB ThinNlight   
455 256 GB 1024 GB Windows 64-bit 0 GB ThinNlight   
629 256 GB 0 GB Windows 64-bit 0 GB ThinNlight   
631 256 GB 0 GB Windows 64-bit 0 GB Casual   
650 512 GB 0 GB Windows 64-bit 0 GB ThinNlight   
655 0 GB 1024 GB Windows 64-bit 0 GB ThinNlight   
656 0 GB 1024 GB Windows 64-bit 0 GB Casual   
657 0 GB 1024 GB Windows 64-bit 0 GB Casual   
667 256 GB 0 GB Windows 64-bit 0 GB ThinNlight   
698 512 GB 0 GB Windows 64-bit 0 GB ThinNlight   
699 512 GB 0 GB Windows 64-bit 6 GB Casual   
710 512 GB 0 GB Windows 64-bit 4 GB Casual   
721 512 GB 0 GB Windows 64-bit 0 GB Casual   
726 1024 GB 0 GB Windows 32-bit 0 GB Casual   
736 256 GB 1024 GB Windows 32-bit 0 GB Casual   
764 512 GB 0 GB Windows 64-bit 6 GB Casual   
769 512 GB 0 GB Windows 64-bit 0 GB Casual   
  
 warranty Touchscreen msoffice Price rating Number of Ratings \  
40 1 year No Yes 36990 4 stars 1178   
44 1 year No Yes 39490 4 stars 634   
59 No warranty No No 44427 3 stars 0   
63 1 year No No 42490 3 stars 0   
65 No warranty Yes No 71999 4 stars 97   
76 No warranty No No 43900 3 stars 0   
77 1 year No No 46690 3 stars 0   
79 1 year No No 39990 4 stars 44   
126 No warranty No No 41790 3 stars 0   
141 1 year No No 39990 3 stars 15   
142 No warranty No No 40890 4 stars 527   
146 No warranty No No 43970 3 stars 16   
152 1 year No Yes 43400 4 stars 231   
155 1 year No Yes 43000 4 stars 313   
175 1 year No Yes 42990 4 stars 62   
178 1 year No Yes 43590 3 stars 0   
179 1 year No No 44690 4 stars 9   
181 1 year No Yes 43990 3 stars 0   
182 1 year No Yes 43590 3 stars 0   
183 1 year No Yes 41390 4 stars 56   
184 1 year No Yes 44290 3 stars 0   
185 1 year No No 48999 3 stars 0   
186 1 year No Yes 39990 3 stars 170   
455 1 year No Yes 55790 4 stars 41   
629 1 year No No 36000 4 stars 102   
631 1 year No No 35990 3 stars 0   
650 No warranty No No 40590 3 stars 0   
655 1 year No Yes 52632 3 stars 0   
656 1 year No Yes 39990 4 stars 409   
657 1 year No No 43990 3 stars 0   
667 1 year No No 44390 4 stars 276   
698 1 year No Yes 60990 4 stars 183   
699 No warranty No No 76190 4 stars 358   
710 1 year No No 98920 3 stars 0   
721 1 year Yes Yes 75500 3 stars 0   
726 No warranty No No 57900 4 stars 3   
736 No warranty No No 51990 3 stars 0   
764 No warranty No No 89990 4 stars 191   
769 1 year Yes Yes 89570 3 stars 0   
  
 Number of Reviews   
40 137   
44 64   
59 0   
63 0   
65 13   
76 0   
77 0   
79 2   
126 0   
141 1   
142 46   
146 0   
152 26   
155 35   
175 7   
178 0   
179 0   
181 0   
182 0   
183 7   
184 0   
185 0   
186 11   
455 4   
629 16   
631 0   
650 0   
655 0   
656 39   
657 0   
667 24   
698 42   
699 64   
710 0   
721 0   
726 1   
736 0   
764 27   
769 0

ram\_group = df.groupby('ram\_gb')

pd.DataFrame(ram\_group['Price'].mean())

Price  
ram\_gb   
16 GB 119909.406780  
32 GB 239990.000000  
4 GB 60080.991150  
8 GB 66696.248804

age = owner\_group['age'].agg(mini\_age='min',max\_age='max',mean\_age='mean')  
age

mini\_age max\_age mean\_age  
owner   
First Owner 1.0 43.0 7.684855  
Fourth Owner Or More 4.0 37.0 15.250000  
Second Owner 2.0 63.0 12.129095  
Third Owner 4.0 43.0 11.175926

ram\_group.agg({'Price':'min','Number of Ratings':'mean'})

Price Number of Ratings  
ram\_gb   
16 GB 56990 79.536723  
32 GB 169990 1.500000  
4 GB 16990 401.345133  
8 GB 32490 370.114833

# df[df['os']=='Windows']  
  
os\_list = ['Mac','DOS']  
  
df[df['os'].isin(os\_list)]

brand processor\_brand processor\_name processor\_gnrtn ram\_gb ram\_type \  
198 acer Intel Core i5 10th 8 GB LPDDR4   
199 ASUS Intel Core i5 10th 8 GB LPDDR4   
200 DELL Intel Core i5 10th 8 GB LPDDR4   
201 DELL Intel Core i5 10th 8 GB LPDDR4   
202 HP Intel Core i5 10th 8 GB LPDDR4   
203 HP Intel Core i5 10th 8 GB LPDDR4   
204 Lenovo Intel Core i5 10th 8 GB LPDDR4   
309 APPLE Intel Core i5 10th 8 GB LPDDR4X   
310 APPLE Intel Core i5 8th 8 GB DDR3   
407 APPLE Intel Core i5 10th 16 GB LPDDR4X   
408 APPLE Intel Core i5 10th 16 GB LPDDR4X   
433 MSI Intel Core i5 10th 8 GB LPDDR4   
434 MSI Intel Core i5 10th 8 GB LPDDR4   
435 MSI Intel Core i5 10th 8 GB LPDDR4   
477 MSI Intel Core i5 10th 8 GB LPDDR4   
595 MSI Intel Core i5 10th 8 GB LPDDR4   
596 MSI Intel Core i5 10th 8 GB LPDDR4   
597 MSI Intel Core i5 10th 8 GB LPDDR4   
598 ASUS Intel Core i7 10th 16 GB LPDDR4X   
599 APPLE M1 M1 10th 8 GB DDR4   
600 APPLE M1 M1 10th 8 GB DDR4   
601 APPLE M1 M1 10th 16 GB DDR4   
602 APPLE M1 M1 10th 8 GB DDR4   
603 APPLE M1 M1 10th 16 GB DDR4   
604 APPLE M1 M1 10th 16 GB DDR4   
605 APPLE M1 M1 10th 8 GB DDR4   
606 APPLE M1 M1 10th 16 GB DDR4   
607 APPLE M1 M1 10th 8 GB DDR4   
608 APPLE M1 M1 10th 8 GB DDR4   
609 APPLE M1 M1 10th 8 GB DDR4   
610 APPLE M1 M1 10th 8 GB DDR4   
611 APPLE M1 M1 10th 8 GB DDR4   
612 APPLE M1 M1 10th 16 GB DDR4   
613 APPLE M1 M1 10th 16 GB DDR4   
614 APPLE M1 M1 10th 16 GB DDR4   
615 APPLE M1 M1 10th 4 GB DDR4   
616 APPLE M1 M1 10th 8 GB DDR4   
617 APPLE M1 M1 10th 8 GB DDR4   
618 APPLE M1 M1 10th 16 GB DDR4   
619 APPLE M1 M1 10th 16 GB DDR4   
620 APPLE M1 M1 10th 16 GB DDR4   
621 APPLE M1 M1 10th 32 GB DDR4   
622 APPLE M1 M1 10th 16 GB DDR4   
822 Lenovo AMD Ryzen 5 10th 8 GB DDR4   
  
 ssd hdd os os\_bit graphic\_card\_gb weight warranty \  
198 0 GB 512 GB DOS 32-bit 0 GB Casual No warranty   
199 0 GB 512 GB DOS 32-bit 4 GB Casual No warranty   
200 0 GB 512 GB DOS 32-bit 4 GB Casual 1 year   
201 0 GB 512 GB DOS 32-bit 0 GB ThinNlight 1 year   
202 0 GB 512 GB DOS 32-bit 0 GB ThinNlight No warranty   
203 0 GB 512 GB DOS 32-bit 4 GB Casual No warranty   
204 0 GB 512 GB DOS 32-bit 2 GB Casual No warranty   
309 512 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
310 512 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
407 1024 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
408 1024 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
433 0 GB 512 GB DOS 32-bit 6 GB Casual No warranty   
434 0 GB 512 GB DOS 32-bit 6 GB Casual No warranty   
435 0 GB 512 GB DOS 32-bit 8 GB Casual No warranty   
477 0 GB 512 GB DOS 32-bit 8 GB Casual No warranty   
595 0 GB 512 GB DOS 32-bit 4 GB Casual No warranty   
596 0 GB 512 GB DOS 32-bit 6 GB Casual No warranty   
597 0 GB 512 GB DOS 32-bit 4 GB Casual No warranty   
598 0 GB 512 GB DOS 32-bit 6 GB Casual No warranty   
599 256 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
600 256 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
601 256 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
602 256 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
603 256 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
604 256 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
605 256 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
606 512 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
607 512 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
608 512 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
609 512 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
610 512 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
611 512 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
612 512 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
613 512 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
614 512 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
615 512 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
616 512 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
617 1024 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
618 1024 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
619 1024 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
620 1024 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
621 1024 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
622 1024 GB 0 GB Mac 64-bit 0 GB Casual 1 year   
822 512 GB 0 GB DOS 64-bit 0 GB ThinNlight No warranty   
  
 Touchscreen msoffice Price rating Number of Ratings \  
198 No No 94190 4 stars 31   
199 No No 106167 4 stars 1881   
200 No No 68090 1 star 23   
201 No No 139990 3 stars 0   
202 No No 55990 4 stars 818   
203 No No 102551 3 stars 0   
204 No No 46990 4 stars 4428   
309 No No 159990 4 stars 32   
310 No No 172090 4 stars 108   
407 No No 179990 4 stars 31   
408 No No 179990 4 stars 91   
433 No No 291190 4 stars 7   
434 No No 145990 4 stars 715   
435 No No 179990 4 stars 24   
477 No No 207290 3 stars 0   
595 No No 86990 4 stars 218   
596 No No 109990 4 stars 78   
597 No No 109590 4 stars 5   
598 No No 146490 3 stars 2   
599 No No 85990 4 stars 6725   
600 No No 112990 4 stars 640   
601 No No 104990 4 stars 45   
602 No No 112990 4 stars 640   
603 No No 132990 4 stars 640   
604 No No 104990 4 stars 45   
605 No No 85990 4 stars 6725   
606 No No 149990 4 stars 640   
607 No No 125990 4 stars 399   
608 No No 103990 4 stars 809   
609 No No 101990 4 stars 809   
610 No No 104990 4 stars 6725   
611 No No 131990 4 stars 399   
612 No No 128990 4 stars 45   
613 No No 184990 4 stars 42   
614 No No 225990 4 stars 42   
615 No No 124990 4 stars 45   
616 No No 104990 4 stars 6725   
617 No No 149990 4 stars 640   
618 No No 169990 4 stars 640   
619 No No 225990 4 stars 42   
620 No No 243990 4 stars 42   
621 No No 309990 5 stars 3   
622 No No 225990 4 stars 42   
822 No No 57490 4 stars 18   
  
 Number of Reviews   
198 6   
199 241   
200 4   
201 0   
202 109   
203 0   
204 569   
309 5   
310 13   
407 3   
408 5   
433 3   
434 106   
435 5   
477 0   
595 27   
596 16   
597 2   
598 1   
599 629   
600 70   
601 4   
602 70   
603 70   
604 4   
605 629   
606 70   
607 45   
608 91   
609 91   
610 629   
611 45   
612 4   
613 8   
614 8   
615 4   
616 629   
617 70   
618 70   
619 8   
620 8   
621 1   
622 8   
822 4

df[df['os'].str.contains('Wind')]

brand processor\_brand processor\_name processor\_gnrtn ram\_gb ram\_type \  
0 ASUS Intel Core i3 10th 4 GB DDR4   
1 Lenovo Intel Core i3 10th 4 GB DDR4   
2 Lenovo Intel Core i3 10th 4 GB DDR4   
3 ASUS Intel Core i5 10th 8 GB DDR4   
4 ASUS Intel Celeron Dual Not Available 4 GB DDR4   
.. ... ... ... ... ... ...   
817 ASUS AMD Ryzen 9 Not Available 4 GB LPDDR4X   
818 ASUS AMD Ryzen 9 Not Available 4 GB DDR4   
819 ASUS AMD Ryzen 9 Not Available 4 GB DDR4   
820 ASUS AMD Ryzen 9 Not Available 4 GB DDR4   
821 ASUS AMD Ryzen 9 Not Available 4 GB DDR4   
  
 ssd hdd os os\_bit graphic\_card\_gb weight warranty \  
0 0 GB 1024 GB Windows 64-bit 0 GB Casual No warranty   
1 0 GB 1024 GB Windows 64-bit 0 GB Casual No warranty   
2 0 GB 1024 GB Windows 64-bit 0 GB Casual No warranty   
3 512 GB 0 GB Windows 32-bit 2 GB Casual No warranty   
4 0 GB 512 GB Windows 64-bit 0 GB Casual No warranty   
.. ... ... ... ... ... ... ...   
817 1024 GB 0 GB Windows 64-bit 4 GB Casual 1 year   
818 1024 GB 0 GB Windows 64-bit 0 GB Casual 1 year   
819 1024 GB 0 GB Windows 64-bit 0 GB Casual 1 year   
820 1024 GB 0 GB Windows 64-bit 4 GB Casual 1 year   
821 1024 GB 0 GB Windows 64-bit 4 GB Casual 1 year   
  
 Touchscreen msoffice Price rating Number of Ratings \  
0 No No 34649 2 stars 3   
1 No No 38999 3 stars 65   
2 No No 39999 3 stars 8   
3 No No 69990 3 stars 0   
4 No No 26990 3 stars 0   
.. ... ... ... ... ...   
817 Yes Yes 234990 3 stars 0   
818 No No 135990 3 stars 0   
819 No No 144990 3 stars 0   
820 No No 149990 3 stars 0   
821 No No 142990 3 stars 0   
  
 Number of Reviews   
0 0   
1 5   
2 1   
3 0   
4 0   
.. ...   
817 0   
818 0   
819 0   
820 0   
821 0   
  
[779 rows x 19 columns]

# df.query("os=='Windows'")  
df.query("os\_bit=='64-bit'")

brand processor\_brand processor\_name processor\_gnrtn ram\_gb ram\_type \  
0 ASUS Intel Core i3 10th 4 GB DDR4   
1 Lenovo Intel Core i3 10th 4 GB DDR4   
2 Lenovo Intel Core i3 10th 4 GB DDR4   
4 ASUS Intel Celeron Dual Not Available 4 GB DDR4   
5 ASUS Intel Celeron Dual Not Available 4 GB DDR4   
.. ... ... ... ... ... ...   
818 ASUS AMD Ryzen 9 Not Available 4 GB DDR4   
819 ASUS AMD Ryzen 9 Not Available 4 GB DDR4   
820 ASUS AMD Ryzen 9 Not Available 4 GB DDR4   
821 ASUS AMD Ryzen 9 Not Available 4 GB DDR4   
822 Lenovo AMD Ryzen 5 10th 8 GB DDR4   
  
 ssd hdd os os\_bit graphic\_card\_gb weight \  
0 0 GB 1024 GB Windows 64-bit 0 GB Casual   
1 0 GB 1024 GB Windows 64-bit 0 GB Casual   
2 0 GB 1024 GB Windows 64-bit 0 GB Casual   
4 0 GB 512 GB Windows 64-bit 0 GB Casual   
5 0 GB 512 GB Windows 64-bit 0 GB Casual   
.. ... ... ... ... ... ...   
818 1024 GB 0 GB Windows 64-bit 0 GB Casual   
819 1024 GB 0 GB Windows 64-bit 0 GB Casual   
820 1024 GB 0 GB Windows 64-bit 4 GB Casual   
821 1024 GB 0 GB Windows 64-bit 4 GB Casual   
822 512 GB 0 GB DOS 64-bit 0 GB ThinNlight   
  
 warranty Touchscreen msoffice Price rating Number of Ratings \  
0 No warranty No No 34649 2 stars 3   
1 No warranty No No 38999 3 stars 65   
2 No warranty No No 39999 3 stars 8   
4 No warranty No No 26990 3 stars 0   
5 No warranty No No 22990 3 stars 0   
.. ... ... ... ... ... ...   
818 1 year No No 135990 3 stars 0   
819 1 year No No 144990 3 stars 0   
820 1 year No No 149990 3 stars 0   
821 1 year No No 142990 3 stars 0   
822 No warranty No No 57490 4 stars 18   
  
 Number of Reviews   
0 0   
1 5   
2 1   
4 0   
5 0   
.. ...   
818 0   
819 0   
820 0   
821 0   
822 4   
  
[714 rows x 19 columns]

df['Touchscreen'] = df['Touchscreen'].map({'No':0,'Yes':1})

df['Touchscreen'].value\_counts()

Touchscreen  
0 726  
1 97  
Name: count, dtype: int64

df['ram\_gb'].value\_counts()

ram\_gb  
8 GB 418  
4 GB 226  
16 GB 177  
32 GB 2  
Name: count, dtype: int64

df['ram\_gb'].map(lambda x:x.split(' ')[0])

0 4  
1 4  
2 4  
3 8  
4 4  
 ..  
818 4  
819 4  
820 4  
821 4  
822 8  
Name: ram\_gb, Length: 823, dtype: object

import pandas as pd

df = pd.read\_csv("https://raw.githubusercontent.com/Ranjit-Singh-786/Book-Recommedation\_Project/master/Data/Books/Books.csv")

C:\Users\Jai\AppData\Local\Temp\ipykernel\_8052\4000385519.py:1: DtypeWarning: Columns (3) have mixed types. Specify dtype option on import or set low\_memory=False.  
 df = pd.read\_csv("https://raw.githubusercontent.com/Ranjit-Singh-786/Book-Recommedation\_Project/master/Data/Books/Books.csv")

df.to\_csv('book\_data.csv',index=False)

df

ISBN Book-Title \  
0 0195153448 Classical Mythology   
1 0002005018 Clara Callan   
2 0060973129 Decision in Normandy   
3 0374157065 Flu: The Story of the Great Influenza Pandemic...   
4 0393045218 The Mummies of Urumchi   
... ... ...   
271355 0440400988 There's a Bat in Bunk Five   
271356 0525447644 From One to One Hundred   
271357 006008667X Lily Dale : The True Story of the Town that Ta...   
271358 0192126040 Republic (World's Classics)   
271359 0767409752 A Guided Tour of Rene Descartes' Meditations o...   
  
 Book-Author Year-Of-Publication \  
0 Mark P. O. Morford 2002   
1 Richard Bruce Wright 2001   
2 Carlo D'Este 1991   
3 Gina Bari Kolata 1999   
4 E. J. W. Barber 1999   
... ... ...   
271355 Paula Danziger 1988   
271356 Teri Sloat 1991   
271357 Christine Wicker 2004   
271358 Plato 1996   
271359 Christopher Biffle 2000   
  
 Publisher \  
0 Oxford University Press   
1 HarperFlamingo Canada   
2 HarperPerennial   
3 Farrar Straus Giroux   
4 W. W. Norton &amp; Company   
... ...   
271355 Random House Childrens Pub (Mm)   
271356 Dutton Books   
271357 HarperSanFrancisco   
271358 Oxford University Press   
271359 McGraw-Hill Humanities/Social Sciences/Languages   
  
 Image-URL-S \  
0 http://images.amazon.com/images/P/0195153448.0...   
1 http://images.amazon.com/images/P/0002005018.0...   
2 http://images.amazon.com/images/P/0060973129.0...   
3 http://images.amazon.com/images/P/0374157065.0...   
4 http://images.amazon.com/images/P/0393045218.0...   
... ...   
271355 http://images.amazon.com/images/P/0440400988.0...   
271356 http://images.amazon.com/images/P/0525447644.0...   
271357 http://images.amazon.com/images/P/006008667X.0...   
271358 http://images.amazon.com/images/P/0192126040.0...   
271359 http://images.amazon.com/images/P/0767409752.0...   
  
 Image-URL-M \  
0 http://images.amazon.com/images/P/0195153448.0...   
1 http://images.amazon.com/images/P/0002005018.0...   
2 http://images.amazon.com/images/P/0060973129.0...   
3 http://images.amazon.com/images/P/0374157065.0...   
4 http://images.amazon.com/images/P/0393045218.0...   
... ...   
271355 http://images.amazon.com/images/P/0440400988.0...   
271356 http://images.amazon.com/images/P/0525447644.0...   
271357 http://images.amazon.com/images/P/006008667X.0...   
271358 http://images.amazon.com/images/P/0192126040.0...   
271359 http://images.amazon.com/images/P/0767409752.0...   
  
 Image-URL-L   
0 http://images.amazon.com/images/P/0195153448.0...   
1 http://images.amazon.com/images/P/0002005018.0...   
2 http://images.amazon.com/images/P/0060973129.0...   
3 http://images.amazon.com/images/P/0374157065.0...   
4 http://images.amazon.com/images/P/0393045218.0...   
... ...   
271355 http://images.amazon.com/images/P/0440400988.0...   
271356 http://images.amazon.com/images/P/0525447644.0...   
271357 http://images.amazon.com/images/P/006008667X.0...   
271358 http://images.amazon.com/images/P/0192126040.0...   
271359 http://images.amazon.com/images/P/0767409752.0...   
  
[271360 rows x 8 columns]

df['Image-URL-L'][6:20]

6 http://images.amazon.com/images/P/0425176428.0...  
7 http://images.amazon.com/images/P/0671870432.0...  
8 http://images.amazon.com/images/P/0679425608.0...  
9 http://images.amazon.com/images/P/074322678X.0...  
10 http://images.amazon.com/images/P/0771074670.0...  
11 http://images.amazon.com/images/P/080652121X.0...  
12 http://images.amazon.com/images/P/0887841740.0...  
13 http://images.amazon.com/images/P/1552041778.0...  
14 http://images.amazon.com/images/P/1558746218.0...  
15 http://images.amazon.com/images/P/1567407781.0...  
16 http://images.amazon.com/images/P/1575663937.0...  
17 http://images.amazon.com/images/P/1881320189.0...  
18 http://images.amazon.com/images/P/0440234743.0...  
19 http://images.amazon.com/images/P/0452264464.0...  
Name: Image-URL-L, dtype: object

# top 50 images download

(271360, 8)

df['Image-URL-L'][3]

'http://images.amazon.com/images/P/0374157065.01.LZZZZZZZ.jpg'