pandas-basic

August 8, 2024

1 Learn about pandas.

```
Pandas deals with the data analyse, data manipulation,transformation etc.. operation with da
```

Import pandas

Check version

Generate dataframe

Print dataframe

Acess dataframe rows and columns

Access with loc and iloc

[1]:

```
import numpy as np
import pandas as pd
print(pd.__version__)
```

```
2.2.2
```

[2]:

```
data1={"Name":["rahul","nisha","paras","kaliya"],
      "Branch":["AI&DS","AI&DS","CSE","POGO"],
      "College":["ACE","ACE","JEC","BHEEM"],
      "roll_no":[10,12,56,23]}
data1   # Dictionary is also like a data structure
```

[2]:

```
{'Name': ['rahul', 'nisha', 'paras', 'kaliya'],
 'Branch': ['AI&DS', 'AI&DS', 'CSE', 'POGO'],
 'College': ['ACE', 'ACE', 'JEC', 'BHEEM'],
 'roll_no': [10, 12, 56, 23]}
```

[3]:

```
df1=pd.DataFrame(data1)    # Make dataframe of data
df1
```

[3]:

```
     Name Branch College  roll_no
0   rahul  AI&DS     ACE       10
1   nisha  AI&DS     ACE       12
2   paras    CSE     JEC       56
3  kaliya   POGO   BHEEM       23
```

```
[4]: data = {"Name":["sachin",'lakshay','saurabh','abhishek'],
            "Branch":["cse",'it','ece','cse'],
            "college":['piet','jecrc','raffels','piet'],
            "roll_no":[10,20,30,50]}
     data
```

```
[4]: {'Name': ['sachin', 'lakshay', 'saurabh', 'abhishek'],
      'Branch': ['cse', 'it', 'ece', 'cse'],
      'college': ['piet', 'jecrc', 'raffels', 'piet'],
      'roll_no': [10, 20, 30, 50]}
```

```
[5]: df=pd.DataFrame(data)    # Make dataframe of data
     df
```

```
[5]:        Name Branch  college  roll_no
     0    sachin    cse     piet       10
     1   lakshay     it    jecrc       20
     2   saurabh    ece  raffels       30
     3  abhishek    cse     piet       50
```

```
[6]: type(df)   # the type to confirmation
```

```
[6]: pandas.core.frame.DataFrame
```

```
[7]: df['Name'] #access a single column
```

```
[7]: 0      sachin
     1     lakshay
     2     saurabh
     3    abhishek
     Name: Name, dtype: object
```

```
[8]: type(df['Name'])  # column is known as series
```

```
[8]: pandas.core.series.Series
```

```
[9]: df.roll_no  # another way to accessa single column
```

```
[9]: 0    10
     1    20
     2    30
     3    50
     Name: roll_no, dtype: int64
```

```
[10]: df[["Name","Branch"]]  # to access multiple columns
```

```
[10]:        Name Branch
      0    sachin    cse
      1   lakshay     it
      2   saurabh    ece
      3  abhishek    cse
```

```
[11]: df[["college","roll_no","college"]]  # it shows order dosen't follow . It␣
       ↪depends on which is accessing first . Here college is written first so it␣
       ↪will show first
```

```
[11]:    college  roll_no  college
      0     piet       10     piet
      1    jecrc       20    jecrc
      2  raffels       30  raffels
      3     piet       50     piet
```

1.0.1 Loc and iloc = use to access perticular data from dataframe

Use of iloc = iloc support both row and collumn indexing as numpy

```
[12]: df.iloc[2:,2:]
```

```
[12]:    college  roll_no
      2  raffels       30
      3     piet       50
```

```
[13]: df.iloc[-2:,-2:]  # same can be done with -ve indexing
```

```
[13]:    college  roll_no
      2  raffels       30
      3     piet       50
```

Use of loc

loc dosent support collumn indexing

```
[21]: df.loc[2:3,["college",'roll_no']]
```

```
[21]:    college  roll_no
      2  raffels       30
      3     piet       50
```

```
[15]: df.loc[0:2] # access rows . But according to simple indexing for [0:2] , only 0␣
       ↪or 1 rows should be print but here 3 rows (0,1,2) are generating because in␣
       ↪loc there is inclusive point and no need to access extra . It reaches and␣
       ↪access as the given indexing without leaving last.
```

```
[15]:       Name Branch  college  roll_no
      0   sachin    cse     piet       10
      1  lakshay     it    jecrc       20
      2  saurabh    ece  raffels       30
```

```
[16]: df.loc[0:1]
```

```
[16]:       Name Branch college  roll_no
      0   sachin    cse    piet       10
      1  lakshay     it   jecrc       20
```

Examples of accessing elements of df by loc and iloc with +ve and -ve indexing

```
[17]: df.iloc[1:3,0:3]
```

```
[17]:       Name Branch  college
      1  lakshay     it    jecrc
      2  saurabh    ece  raffels
```

```
[18]: df.iloc[-3:-1,-4:-1]
```

```
[18]:       Name Branch  college
      1  lakshay     it    jecrc
      2  saurabh    ece  raffels
```

```
[19]: df.loc[1:2,["Name","Branch","college"]]
```

```
[19]:       Name Branch  college
      1  lakshay     it    jecrc
      2  saurabh    ece  raffels
```

```
[20]: df.loc[-3:-2,["Name","Branch","college"]]
```

```
[20]: Empty DataFrame
      Columns: [Name, Branch, college]
      Index: []
```

pandas-excel-file

August 8, 2024

1 PANDAS (EXCEL)

Import excel file

Different type of functions and operations

`(Head , Tail , Info , Dtype , Shape , Duplicated , Drop_duplicated)`

Searching and selecting operations

filtering

drop column

[369]:
```python
import pandas as pd
```

[370]:
```python
df=pd.read_csv("Used_Bikes.csv")    #file can also be accessed with file path
df
```

[370]:
```
                                  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]
```

[371]:
```
df.head()    # return the top head rows
```

[371]:
```
                              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  
```

[372]:
```
df.head(10)     # as want give value to that no rows
```

[372]:
```
                                  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
```

[373]:

```
df.tail()   # acess the bottom rows
```

[373]:

```
                       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
```

[374]:

```
df.tail(12) #access the specific no of bottom rows
```

[374]:

```
                         bike_name     price         city  kms_driven  \
32636                KTM RC 390cc  196700.0       Mumbai     13216.0
32637           Bajaj Pulsar 150cc   25000.0        Delhi     32588.0
32638        Yamaha Fazer 25 250cc  123000.0       Kadapa     14500.0
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
32642           Hero Passion 100cc   15000.0  Perumbavoor     35000.0
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
32636   First Owner   4.0  390.0            KTM
32637   First Owner   9.0  150.0          Bajaj
32638   First Owner   4.0  250.0         Yamaha
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
32642  Second Owner  19.0  100.0           Hero
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
```

```
[375]: df.info() # give the short information about dataframe
```

```
<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
```

```
[376]: df.dtypes   # datatype ca be checked by .dtypes or .info
```

```
[376]: bike_name      object
       price         float64
       city           object
       kms_driven    float64
       owner          object
       age           float64
       power         float64
       brand          object
       dtype: object
```

```
[377]: df["age"].dtype   # to get the datatype of a perticular column
```

```
[377]: dtype('float64')
```

```
[378]: df.shape    # is shows rows and column as like matrix
```

```
[378]: (32648, 8)
```

```
[379]: df.duplicated()  # check there is duplicate or not?
```

```
[379]: 0        False
       1        False
       2        False
       3        False
       4        False
                …
       32643     True
```

```
32644    True
32645    True
32646    True
32647    True
Length: 32648, dtype: bool
```

[380]:
```
df.duplicated().sum()   # check the total no of duplicates
```

[380]:
```
np.int64(25324)
```

[381]:
```
df.drop_duplicates()  # to remove the duplicates
```

[381]:
```
                                 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]
```

[382]:
```
df   #is shows the complete data again
```

[382]:
```
                                 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]
```

[383]:

```
# to remove duplicate permanent
df.drop_duplicates(inplace=True)
```

[384]:

```
df   #Updated data
```

[384]:

```
                                 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]
```

```
[385]: df.shape   # Now shape is changed because duplicates are removed
```

```
[385]: (7324, 8)
```

```
[386]: df['brand'] # access column
```

```
[386]: 0                    TVS
       1          Royal Enfield
       2                Triumph
       3                    TVS
       4                 Yamaha
                    …
       9362               Hero
       9369              Bajaj
       9370    Harley-Davidson
       9371              Bajaj
       9372              Bajaj
       Name: brand, Length: 7324, dtype: object
```

```
[387]: df['brand'].nunique()  # nunique (number of unique)to check how many brands are␣
        ↪unique ( no copy )
```

```
[387]: 23
```

```
[388]: df['brand'].unique() # total unique brands
```

```
[388]: array(['TVS', 'Royal Enfield', 'Triumph', 'Yamaha', 'Honda', 'Hero',
              'Bajaj', 'Suzuki', 'Benelli', 'KTM', 'Mahindra', 'Kawasaki',
              'Ducati', 'Hyosung', 'Harley-Davidson', 'Jawa', 'BMW', 'Indian',
              'Rajdoot', 'LML', 'Yezdi', 'MV', 'Ideal'], dtype=object)
```

```
[389]: df["brand"].value_counts()   # values of every brand . It looks like a ␣
        ↪dictionary in which brnad name is like a key and storing values.
```

```
[389]: 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
```

[390]:
```
df["brand"].value_counts().keys()   # It retuen all the keys from brand
```

[390]:
```
Index(['Bajaj', 'Royal Enfield', 'Hero', 'Honda', 'Yamaha', 'TVS', 'KTM',
       'Suzuki', 'Harley-Davidson', 'Kawasaki', 'Hyosung', 'Mahindra',
       'Benelli', 'Triumph', 'Ducati', 'BMW', 'Jawa', 'Indian', 'MV',
       'Rajdoot', 'LML', 'Yezdi', 'Ideal'],
      dtype='object', name='brand')
```

[391]:
```
[df["brand"].value_counts().values]   # It retuen all the values according to␣
 ↪key order from brand
```

[391]:
```
[array([2081, 1346, 1142,  676,  651,  481,  375,  203,   91,   61,   53,
          50,   46,   21,   20,   10,    7,    3,    3,    1,    1,    1,
           1])]
```

[392]:
```
df["city"].value_counts()  # count of every city
```

[392]:
```
city
Delhi          1426
Bangalore       683
Mumbai          609
Gurgaon         474
Faridabad       463
               …
Berhampore        1
```

```
Silvasa          1
Hospet           1
Palai            1
Sidhi            1
Name: count, Length: 443, dtype: int64
```

[393]:
```
df["city"].value_counts().head(4)    # count of top 4 city
```

[393]:
```
city
Delhi        1426
Bangalore     683
Mumbai        609
Gurgaon       474
Name: count, dtype: int64
```

1.1 Filtering operations

[394]:
```
bullet=df[df["brand"]=="Royal Enfield"]    #find data of royal enfield , data fo␣
  ↪royal enfield store in bullet varaible
bullet                                     # bullet is part of df so df is␣
  ↪called population data and bullet called sample data
```

[394]:
```
                                     bike_name     price       city  \
1                  Royal Enfield Classic 350cc  119900.0      Delhi
8            Royal Enfield Thunderbird X 350cc  145000.0  Bangalore
9     Royal Enfield Classic Desert Storm 500cc   88000.0      Delhi
23          Royal Enfield Classic Chrome 500cc  121700.0     Kalyan
36                 Royal Enfield Classic 350cc   98800.0      Kochi
…                                        …       …        …
9261               Royal Enfield Classic 500cc  146006.0   Guwahati
9319               Royal Enfield Classic 350cc  100000.0    Chennai
9337             Royal Enfield Himalayan 410cc  120000.0    Gurgaon
9338             Royal Enfield Himalayan 410cc  138000.0      Delhi
9344      Royal Enfield Bullet Twinspark 350cc   80000.0      Delhi

      kms_driven         owner   age  power          brand
1        11000.0   First Owner   4.0  350.0  Royal Enfield
8         9190.0   First Owner   3.0  350.0  Royal Enfield
9        19000.0  Second Owner   7.0  500.0  Royal Enfield
23       24520.0   First Owner   5.0  500.0  Royal Enfield
36       39000.0   First Owner   5.0  350.0  Royal Enfield
…          …           …   …    …            …
9261      8575.0   First Owner   4.0  500.0  Royal Enfield
9319     25000.0   First Owner  10.0  350.0  Royal Enfield
9337      8492.0   First Owner   5.0  410.0  Royal Enfield
9338      5000.0   First Owner   5.0  410.0  Royal Enfield
9344     56968.0   First Owner   8.0  350.0  Royal Enfield
```

```
[1346 rows x 8 columns]
```

```
[395]: df["owner"].nunique()   # show no of unique owners
```

```
[395]: 4
```

```
[396]: df["owner"].unique()   # show unique owners
```

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

```
[397]: df["owner"].value_counts()   # count no of every unique owners
```

```
[397]: owner
        First Owner             6642
        Second Owner             588
        Third Owner               84
        Fourth Owner Or More      10
        Name: count, dtype: int64
```

1.1.1 Acess data having different conditions

(condition1)&(condition2)&(condition3)…

```
[398]: (df["brand"]=="Royal Enfield")&(df["age"]<=4)&(df["owner"]=="first Owner")   #␣
         ↪is shows true or false about indexing nummber
```

```
[398]: 0        False
        1        False
        2        False
        3        False
        4        False
                 …
        9362     False
        9369     False
        9370     False
        9371     False
        9372     False
        Length: 7324, dtype: bool
```

```
[399]: bullet2=df[(df["brand"]=="Royal Enfield")&(df["age"]<=4)&(df["owner"]=="First␣
         ↪Owner")]   # show data of brand = royal enfield include age less then 4 and␣
         ↪of first owner
        bullet2
```

```
[399]:                                  bike_name     price         city  \
        1                  Royal Enfield Classic 350cc  119900.0       Delhi
```

```
8            Royal Enfield Thunderbird X 350cc  145000.0   Bangalore
38           Royal Enfield Thunderbird X 500cc  190500.0  Samastipur
73           Royal Enfield Thunderbird X 350cc  150000.0   Bangalore
77             Royal Enfield Thunderbird 350cc  115000.0   Bangalore
…                                          …         …         …
8825                  Royal Enfield Bullet 350cc  130000.0     Gurgaon
8836      Royal Enfield Thunderbird X 350cc ABS  170200.0      Mumbai
8839  Royal Enfield Classic Desert Storm 500cc  160000.0       Noida
9245                 Royal Enfield Classic 350cc  105000.0       Delhi
9261                 Royal Enfield Classic 500cc  146006.0    Guwahati

      kms_driven        owner  age  power          brand
1        11000.0  First Owner  4.0  350.0  Royal Enfield
8         9190.0  First Owner  3.0  350.0  Royal Enfield
38        4550.0  First Owner  2.0  500.0  Royal Enfield
73       15000.0  First Owner  3.0  350.0  Royal Enfield
77       23700.0  First Owner  4.0  350.0  Royal Enfield
…            …          …    …     …            …
8825     18832.0  First Owner  4.0  350.0  Royal Enfield
8836      1000.0  First Owner  2.0  350.0  Royal Enfield
8839      1754.0  First Owner  4.0  500.0  Royal Enfield
9245     14779.0  First Owner  4.0  350.0  Royal Enfield
9261      8575.0  First Owner  4.0  500.0  Royal Enfield

[388 rows x 8 columns]
```

[400]:
```
bullet2.shape
```

[400]:
```
(388, 8)
```

[401]:
```
bullet3=df[(df["brand"]=="Bajaj")&(df["owner"]=="Second␣
 ↪Owner")&(df["price"]<=65000)&(df["age"]<=3)]
bullet3
```

[401]:
```
                          bike_name    price     city  kms_driven  \
327        Bajaj Avenger Cruise 220  55250.0     Pune      7781.0
5852  Bajaj Avenger Street 220 ABS  45000.0  Chennai     35000.0

             owner  age  power  brand
327   Second Owner  3.0  220.0  Bajaj
5852  Second Owner  2.0  220.0  Bajaj
```

[402]:
```
# to make a seperate file of seperated data
bullet2.to_csv("filter_of_royal_enfield.csv",index=False)    # index = false, ␣
 ↪this is written to leave the indexing according to datframe fetched, and in␣
 ↪new file generate from 0 to no to entries
```

```
[403]: df[df["brand"]=="Bajaj"]    # show data for Bajaj brand
```

```
[403]:                    bike_name     price       city  kms_driven        owner  \
      12          Bajaj Pulsar NS200   78000.0  Bangalore      9900.0  First Owner
      13         Bajaj Discover 100M   29499.0      Delhi     20000.0  First Owner
      14         Bajaj Discover 125M   29900.0      Delhi     20000.0  First Owner
      15      Bajaj Pulsar NS200 ABS   90000.0  Bangalore     11574.0  First Owner
      16      Bajaj Pulsar RS200 ABS  120000.0  Bangalore     23000.0  First Owner
      …                          …        …         …          …           …
      9360        Bajaj Pulsar NS200   48000.0  Allahabad     41939.0  First Owner
      9361       Bajaj Avenger 220cc   50000.0  Bangalore     29134.0  First Owner
      9369       Bajaj Avenger 220cc   35000.0  Bangalore     60000.0  First Owner
      9371     Bajaj Dominar 400 ABS  139000.0  Hyderabad     21300.0  First Owner
      9372  Bajaj Avenger Street 220   80000.0  Hyderabad      7127.0  First Owner

            age  power  brand
      12    4.0  200.0  Bajaj
      13    8.0  100.0  Bajaj
      14    7.0  125.0  Bajaj
      15    3.0  200.0  Bajaj
      16    3.0  200.0  Bajaj
      …     …     …     …
      9360  8.0  200.0  Bajaj
      9361  7.0  220.0  Bajaj
      9369  9.0  220.0  Bajaj
      9371  4.0  400.0  Bajaj
      9372  5.0  220.0  Bajaj

      [2081 rows x 8 columns]
```

```
[404]: df[df["brand"]=="Hero"]
```

```
[404]:                               bike_name    price       city  kms_driven  \
      7     Hero Splendor Plus Self Alloy 100cc  45000.0      Delhi     12645.0
      22    Hero Splendor iSmart Plus IBS 110cc  46500.0      Delhi      3500.0
      26              Hero Super Splendor 125cc  20000.0  Ahmedabad     29305.0
      48                        Hero Hunk 150cc  37000.0     Mumbai     10800.0
      66                   Hero CD Deluxe 100cc  12200.0       Agra     46643.0
      …                                     …        …         …          …
      9316                 Hero Glamour Fi 125cc  37000.0      Delhi     28478.0
      9339             Hero Splendor Plus 100cc  11400.0    Gurgaon     20000.0
      9341                 Hero CD Deluxe 100cc  25000.0      Sidhi     11122.0
      9343               Hero Passion Plus 100cc  24000.0  Hyderabad     68000.0
      9362            Hero Hunk Rear Disc 150cc  25000.0      Delhi     48587.0

                  owner   age  power brand
      7     First Owner   3.0  100.0  Hero
```

```
22     First Owner   2.0  110.0  Hero
26     First Owner  16.0  125.0  Hero
48     First Owner   8.0  150.0  Hero
66     First Owner  14.0  100.0  Hero
…            …   …    …   …
9316   First Owner   5.0  125.0  Hero
9339  Second Owner  17.0  100.0  Hero
9341   First Owner  11.0  100.0  Hero
9343   First Owner  14.0  100.0  Hero
9362   First Owner   8.0  150.0  Hero

[1142 rows x 8 columns]
```

```
[405]: # Retuen data of multiple brands
a=df[(df['brand']=='Bajaj') | (df['brand']=='TVS') | (df['brand']=='Hero') |␣
 ↪(df['brand']=='Yamaha')]   # '|' use of OR operation . this command shoes␣
 ↪inforamtion of given brands
a
```

```
[405]:                                  bike_name     price       city  kms_driven  \
0       TVS Star City Plus Dual Tone 110cc   35000.0  Ahmedabad     17654.0
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
7      Hero Splendor Plus Self Alloy 100cc   45000.0      Delhi     12645.0
…                                    …       …        …         …
9361                   Bajaj Avenger 220cc   50000.0  Bangalore     29134.0
9362             Hero Hunk Rear Disc 150cc   25000.0      Delhi     48587.0
9369                   Bajaj Avenger 220cc   35000.0  Bangalore     60000.0
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
3     First Owner  4.0  180.0     TVS
4     First Owner  3.0  150.0  Yamaha
5     First Owner  6.0  150.0  Yamaha
7     First Owner  3.0  100.0    Hero
…           …  …    …     …
9361  First Owner  7.0  220.0   Bajaj
9362  First Owner  8.0  150.0    Hero
9369  First Owner  9.0  220.0   Bajaj
9371  First Owner  4.0  400.0   Bajaj
9372  First Owner  5.0  220.0   Bajaj

[4355 rows x 8 columns]
```

```
[406]: brands=['Bajaj','TVS','Hero','Yamaha']   # above command is also used as this␣
         ↪to show different brands data using isin function
       df[df["brand"].isin(brands)]
```

```
[406]:                                  bike_name     price       city  kms_driven  \
       0       TVS Star City Plus Dual Tone 110cc   35000.0  Ahmedabad     17654.0
       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
       7      Hero Splendor Plus Self Alloy 100cc   45000.0      Delhi     12645.0
       …                                    …       …        …         …
       9361                   Bajaj Avenger 220cc   50000.0  Bangalore     29134.0
       9362             Hero Hunk Rear Disc 150cc   25000.0      Delhi     48587.0
       9369                   Bajaj Avenger 220cc   35000.0  Bangalore     60000.0
       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
       3     First Owner  4.0  180.0     TVS
       4     First Owner  3.0  150.0  Yamaha
       5     First Owner  6.0  150.0  Yamaha
       7     First Owner  3.0  100.0    Hero
       …           …  …    …     …
       9361  First Owner  7.0  220.0   Bajaj
       9362  First Owner  8.0  150.0    Hero
       9369  First Owner  9.0  220.0   Bajaj
       9371  First Owner  4.0  400.0   Bajaj
       9372  First Owner  5.0  220.0   Bajaj

       [4355 rows x 8 columns]
```

```
[407]: df.drop(["bike_name"],axis="columns")  # remove a column . when inplace = True␣
         ↪given then data permently updated.
       #here inplace= True not given so its only a writing operation
```

```
[407]:          price       city  kms_driven        owner  age  power  \
       0       35000.0  Ahmedabad     17654.0  First Owner  3.0  110.0
       1      119900.0      Delhi     11000.0  First Owner  4.0  350.0
       2      600000.0      Delhi       110.0  First Owner  8.0  675.0
       3       65000.0  Bangalore     16329.0  First Owner  4.0  180.0
       4       80000.0  Bangalore     10000.0  First Owner  3.0  150.0
       …          …        …         …      …  …    …
       9362    25000.0      Delhi     48587.0  First Owner  8.0  150.0
       9369    35000.0  Bangalore     60000.0  First Owner  9.0  220.0
       9370   450000.0    Jodhpur      3430.0  First Owner  4.0  750.0
       9371   139000.0  Hyderabad     21300.0  First Owner  4.0  400.0
```

```
9372   80000.0  Hyderabad      7127.0  First Owner  5.0  220.0

                brand
0                 TVS
1       Royal Enfield
2             Triumph
3                 TVS
4              Yamaha
…               …
9362             Hero
9369            Bajaj
9370  Harley-Davidson
9371            Bajaj
9372            Bajaj

[7324 rows x 7 columns]
```

[408]:
```
df.drop(["bike_name"],axis=1)  # remove a column
# axis=1 for columns
#axis =0 for rows
```

[408]:
```
         price       city  kms_driven        owner  age  power  \
0      35000.0  Ahmedabad     17654.0  First Owner  3.0  110.0
1     119900.0      Delhi     11000.0  First Owner  4.0  350.0
2     600000.0      Delhi       110.0  First Owner  8.0  675.0
3      65000.0  Bangalore     16329.0  First Owner  4.0  180.0
4      80000.0  Bangalore     10000.0  First Owner  3.0  150.0
…        …        …         …          …  …    …
9362   25000.0      Delhi     48587.0  First Owner  8.0  150.0
9369   35000.0  Bangalore     60000.0  First Owner  9.0  220.0
9370  450000.0    Jodhpur      3430.0  First Owner  4.0  750.0
9371  139000.0  Hyderabad     21300.0  First Owner  4.0  400.0
9372   80000.0  Hyderabad      7127.0  First Owner  5.0  220.0

                brand
0                 TVS
1       Royal Enfield
2             Triumph
3                 TVS
4              Yamaha
…               …
9362             Hero
9369            Bajaj
9370  Harley-Davidson
9371            Bajaj
9372            Bajaj
```

```
[7324 rows x 7 columns]
```

[409]:
```
df.drop(["bike_name","owner"],axis="columns") # remove multiple column at a␣
↪time  ( its a writing operation not updating operation)
```

[409]:
```
         price       city  kms_driven  age  power            brand
0      35000.0  Ahmedabad     17654.0  3.0  110.0              TVS
1     119900.0      Delhi     11000.0  4.0  350.0    Royal Enfield
2     600000.0      Delhi       110.0  8.0  675.0          Triumph
3      65000.0  Bangalore     16329.0  4.0  180.0              TVS
4      80000.0  Bangalore     10000.0  3.0  150.0           Yamaha
…        …        …         …  …    …              …
9362   25000.0      Delhi     48587.0  8.0  150.0             Hero
9369   35000.0  Bangalore     60000.0  9.0  220.0            Bajaj
9370  450000.0    Jodhpur      3430.0  4.0  750.0  Harley-Davidson
9371  139000.0  Hyderabad     21300.0  4.0  400.0            Bajaj
9372   80000.0  Hyderabad      7127.0  5.0  220.0            Bajaj

[7324 rows x 6 columns]
```

[410]:
```
df   # df remain same
```

[410]:
```
                                 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]
```

[411]:
```
# to remove permanently
df.drop(["bike_name","owner"],axis="columns",inplace=True)
df
```

[411]:
```
         price       city  kms_driven  age  power            brand
0      35000.0  Ahmedabad     17654.0  3.0  110.0              TVS
1     119900.0      Delhi     11000.0  4.0  350.0    Royal Enfield
2     600000.0      Delhi       110.0  8.0  675.0          Triumph
3      65000.0  Bangalore     16329.0  4.0  180.0              TVS
4      80000.0  Bangalore     10000.0  3.0  150.0           Yamaha
…        …        …         …  …    …               …
9362   25000.0      Delhi     48587.0  8.0  150.0             Hero
9369   35000.0  Bangalore     60000.0  9.0  220.0            Bajaj
9370  450000.0    Jodhpur      3430.0  4.0  750.0  Harley-Davidson
9371  139000.0  Hyderabad     21300.0  4.0  400.0            Bajaj
9372   80000.0  Hyderabad      7127.0  5.0  220.0            Bajaj

[7324 rows x 6 columns]
```

[412]:
```
df.drop("price",axis=1)
```

[412]:
```
           city  kms_driven  age  power            brand
0     Ahmedabad     17654.0  3.0  110.0              TVS
1         Delhi     11000.0  4.0  350.0    Royal Enfield
2         Delhi       110.0  8.0  675.0          Triumph
3     Bangalore     16329.0  4.0  180.0              TVS
4     Bangalore     10000.0  3.0  150.0           Yamaha
…         …         …  …    …               …
9362      Delhi     48587.0  8.0  150.0             Hero
9369  Bangalore     60000.0  9.0  220.0            Bajaj
9370    Jodhpur      3430.0  4.0  750.0  Harley-Davidson
9371  Hyderabad     21300.0  4.0  400.0            Bajaj
9372  Hyderabad      7127.0  5.0  220.0            Bajaj

[7324 rows x 5 columns]
```

pandas-data-hendling

August 8, 2024

1 PANDAS (Data Hendling)

Column generating

Feature engineering * make new column using existing column

- Missing value handling -

(1). Remove missing record

(2). Fill missing value

group and get_group feature

MAP()

[1]:
```
import pandas as pd
import numpy as np
```

[4]:
```
df=pd.read_csv("Used_Bikes.csv")
df.head()
```

[4]:
```
                              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
```

Add column in database

[3]:
```
df["B"]      # it shows key error . because there is no B named column
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File c:\Users\HARSH KUMAR␣
 ↪SAINI\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\indexes\base.
 ↪py:3805, in Index.get_loc(self, key)
   3804 try:
-> 3805     return self._engine.get_loc(casted_key)
   3806 except KeyError as err:

File index.pyx:167, in pandas._libs.index.IndexEngine.get_loc()

File index.pyx:196, in pandas._libs.index.IndexEngine.get_loc()

File pandas\\_libs\\hashtable_class_helper.pxi:7081, in pandas._libs.hashtable.
 ↪PyObjectHashTable.get_item()

File pandas\\_libs\\hashtable_class_helper.pxi:7089, in pandas._libs.hashtable.
 ↪PyObjectHashTable.get_item()

KeyError: 'B'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[3], line 1
----> 1 df["B"]     # it shows key error . because there is no B named column

File c:\Users\HARSH KUMAR␣
 ↪SAINI\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\frame.
 ↪py:4102, in DataFrame.__getitem__(self, key)
   4100 if self.columns.nlevels > 1:
   4101     return self._getitem_multilevel(key)
-> 4102 indexer = self.columns.get_loc(key)
   4103 if is_integer(indexer):
   4104     indexer = [indexer]

File c:\Users\HARSH KUMAR␣
 ↪SAINI\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\indexes\base.
 ↪py:3812, in Index.get_loc(self, key)
   3807     if isinstance(casted_key, slice) or (
   3808         isinstance(casted_key, abc.Iterable)
   3809         and any(isinstance(x, slice) for x in casted_key)
   3810     ):
   3811         raise InvalidIndexError(key)
-> 3812     raise KeyError(key) from err
   3813 except TypeError:
   3814     # If we have a listlike key, _check_indexing_error will raise
   3815     #  InvalidIndexError. Otherwise we fall through and re-raise
   3816     #  the TypeError.
   3817     self._check_indexing_error(key)

KeyError: 'B'
```

[ ]:
```
# Start to making column

df["B"]="Upflairs"    # At the end B named column  is added with scaler value Upflairs
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         B
0       First Owner   3.0  110.0            TVS  Upflairs
1       First Owner   4.0  350.0  Royal Enfield  Upflairs
2       First Owner   8.0  675.0        Triumph  Upflairs
3       First Owner   4.0  180.0            TVS  Upflairs
4       First Owner   3.0  150.0         Yamaha  Upflairs
…                 …     …      …              …         …
32643   First Owner   4.0  100.0           Hero  Upflairs
32644   First Owner   9.0  180.0            TVS  Upflairs
32645   First Owner   6.0  220.0          Bajaj  Upflairs
32646   First Owner  16.0  125.0           Hero  Upflairs
32647   First Owner  13.0  150.0          Bajaj  Upflairs

[32648 rows x 9 columns]
```

1.1 Feature engineering

To make new column using existing column

[ ]:
```
df["Updated_price"]=df["price"]+5000   # by this Updated_price column is generated using price column (Writing operation.. it doesn't effect real data)
```

```
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         B  Updated_price
     0  First Owner  3.0  110.0            TVS  Upflairs        40000.0
     1  First Owner  4.0  350.0  Royal Enfield  Upflairs       124900.0
     2  First Owner  8.0  675.0        Triumph  Upflairs       605000.0
     3  First Owner  4.0  180.0            TVS  Upflairs        70000.0
     4  First Owner  3.0  150.0         Yamaha  Upflairs        85000.0
```

1.1.1 Missing value handling

Example :-

```
[ ]: data = {'A':[10,np.nan,20,50,30,40,np.nan],
            'B':[np.nan,50,60,70,np.nan,25,74],
            'c':[99,50,np.nan,70,np.nan,90,74],
            'D':[1,2,3,4,5,6,7]}                  # np.nana = none space ( not␣
      ↪containing any value )
     data
```

```
[ ]: {'A': [10, nan, 20, 50, 30, 40, nan],
      'B': [nan, 50, 60, 70, nan, 25, 74],
      'c': [99, 50, nan, 70, nan, 90, 74],
      'D': [1, 2, 3, 4, 5, 6, 7]}
```

```
[ ]: df2=pd.DataFrame(data)
     df2                  # naN is missing value
```

```
[ ]:       A     B     c  D
     0  10.0   NaN  99.0  1
     1   NaN  50.0  50.0  2
     2  20.0  60.0   NaN  3
     3  50.0  70.0  70.0  4
     4  30.0   NaN   NaN  5
     5  40.0  25.0  90.0  6
     6   NaN  74.0  74.0  7
```

To find null value(missing)

```
[ ]: df2.isnull()   # true shows there is null value
[ ]:        A      B      c      D
     0  False   True  False  False
     1   True  False  False  False
     2  False  False   True  False
     3  False  False  False  False
     4  False   True   True  False
     5  False  False  False  False
     6   True  False  False  False
```

```
[ ]: df2.isnull().sum()   #count null values collumn wise
```

```
[ ]: A    2
     B    2
     c    2
     D    0
     dtype: int64
```

```
[ ]: df2.isnull().sum().sum()  # count all missing values
```

```
[ ]: np.int64(6)
```

1.1.2 Types to handle missing values

1. Fill the missing value
2. Remove the missing records

Remove the missing reports 2 types - Row wise , Column wise

```
[ ]: df2.dropna()  # this command removes all the records(rows) which has null values
```

```
[ ]:       A     B     c  D
     3  50.0  70.0  70.0  4
     5  40.0  25.0  90.0  6
```

```
[ ]: # it can be written as
     df2.dropna(axis=0)   # axis=0 is used for rows
```

```
[ ]:       A     B     c  D
     3  50.0  70.0  70.0  4
     5  40.0  25.0  90.0  6
```

```
[ ]: df2.dropna(axis="columns")  # this command removes all columns which has missing(null) value
```

```
[ ]:    D
     0  1
     1  2
```

```
2  3
3  4
4  5
5  6
6  7
```

```
[ ]: # it can be written as
     df2.dropna(axis=1)
```

```
[ ]:    D
     0  1
     1  2
     2  3
     3  4
     4  5
     5  6
     6  7
```

1.1.3 Fill the missing values

2 types - * (1) Fill for entirely * (2) Fill for partially

- Fill for entirely

```
[ ]: df2.fillna("Upflairs")   # it fills upfliars at every place of nun value
```

```
[ ]:           A         B         c  D
     0      10.0  Upflairs      99.0  1
     1  Upflairs      50.0      50.0  2
     2      20.0      60.0  Upflairs  3
     3      50.0      70.0      70.0  4
     4      30.0  Upflairs  Upflairs  5
     5      40.0      25.0      90.0  6
     6  Upflairs      74.0      74.0  7
```

- Fill for perticularlly

```
[ ]: df2['B'].fillna("UPFLAIRS")  # fill value for a entire column null space
```

```
[ ]: 0    UPFLAIRS
     1        50.0
     2        60.0
     3        70.0
     4    UPFLAIRS
     5        25.0
     6        74.0
     Name: B, dtype: object
```

column

- categorical -> constant , mode
- numerical -> mean , median

```
[ ]: # mean for numerial column
     df2["c"].fillna(df2["c"].mean()) # when distributon is in normal form then use␣
      ↪mean value
```

```
[ ]: 0    99.0
     1    50.0
     2    76.6
     3    70.0
     4    76.6
     5    90.0
     6    74.0
     Name: c, dtype: float64
```

```
[ ]: # median for numerical column
     df2["c"].fillna(df2["c"].median()) # when distributon is not normal form (␣
      ↪skewed) then use mean value
```

```
[ ]: 0    99.0
     1    50.0
     2    74.0
     3    70.0
     4    74.0
     5    90.0
     6    74.0
     Name: c, dtype: float64
```

```
[ ]: 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]
```

```
[ ]: df5=df[df["brand"]=="Royal Enfield"]
     df5["price"].min()
```

```
[ ]: np.float64(33500.0)
```

1.2 GROUP AND GET_GROUP

```
[8]: # group by operations

     group=df.groupby("brand")
```

```
[10]: GROUP_TVS=group.get_group('TVS')
      GROUP_TVS
```

```
[10]:                                bike_name    price       city  kms_driven  \
      0      TVS Star City Plus Dual Tone 110cc  35000.0  Ahmedabad     17654.0
      3                    TVS Apache RTR 180cc  65000.0  Bangalore     16329.0
      52                   TVS Apache RTR 160cc  60000.0     Mumbai     30000.0
      114          TVS Apache RTR 160 4V Disc  69900.0      Delhi      8700.0
      130                 TVS Phoenix Disc 125cc  21500.0    Barasat     10500.0
      …                                    …      …          …        …
      32549                TVS Apache RTR 180cc  30000.0     Karnal      6639.0
      32568                TVS Apache RTR 180cc  30000.0     Karnal      6639.0
      32587                TVS Apache RTR 180cc  30000.0     Karnal      6639.0
      32606                TVS Apache RTR 180cc  30000.0     Karnal      6639.0
      32644                TVS Apache RTR 180cc  30000.0     Karnal      6639.0

                   owner  age  power brand
      0      First Owner  3.0  110.0   TVS
      3      First Owner  4.0  180.0   TVS
      52     First Owner  5.0  160.0   TVS
      114    First Owner  3.0  160.0   TVS
      130    First Owner  5.0  125.0   TVS
      …              …   …    …     …
      32549  First Owner  9.0  180.0   TVS
      32568  First Owner  9.0  180.0   TVS
```

```
32587  First Owner  9.0  180.0   TVS
32606  First Owner  9.0  180.0   TVS
32644  First Owner  9.0  180.0   TVS

[1247 rows x 8 columns]
```

```
[ ]: group["price"].min()    # access price column of every group and find every of␣
       ↪min()
```

```
[ ]: brand
     BMW                 255000.0
     Bajaj                 6400.0
     Benelli             110700.0
     Ducati              380000.0
     Harley-Davidson     250000.0
     Hero                  5000.0
     Honda                10000.0
     Hyosung             120000.0
     Ideal               100000.0
     Indian              700000.0
     Jawa                146000.0
     KTM                  55000.0
     Kawasaki            110000.0
     LML                   4400.0
     MV                  950000.0
     Mahindra             17800.0
     Rajdoot              75000.0
     Royal Enfield        33500.0
     Suzuki                8000.0
     TVS                   5800.0
     Triumph             500000.0
     Yamaha                9400.0
     Yezdi                68000.0
     Name: price, dtype: float64
```

```
[12]: group[["price"]].min()
```

```
[12]:                    price
      brand
      BMW               255000.0
      Bajaj               6400.0
      Benelli           110700.0
      Ducati            380000.0
      Harley-Davidson   250000.0
      Hero                5000.0
      Honda              10000.0
      Hyosung           120000.0
```

```
Ideal             100000.0
Indian            700000.0
Jawa              146000.0
KTM                55000.0
Kawasaki          110000.0
LML                 4400.0
MV                950000.0
Mahindra           17800.0
Rajdoot            75000.0
Royal Enfield      33500.0
Suzuki              8000.0
TVS                 5800.0
Triumph           500000.0
Yamaha              9400.0
Yezdi              68000.0
```

[ ]:

```
group["price"].agg(min_charges="min",max_cahrges='max',avg_charges='mean')
```

[ ]:

```
                 min_charges  max_cahrges   avg_charges
brand
BMW                 255000.0    1800000.0  5.987500e+05
Bajaj                 6400.0     195000.0  4.833127e+04
Benelli             110700.0     785000.0  2.942000e+05
Ducati              380000.0    1500000.0  9.355455e+05
Harley-Davidson     250000.0    1100000.0  4.529988e+05
Hero                  5000.0     104000.0  2.382945e+04
Honda                10000.0     800000.0  5.923047e+04
Hyosung             120000.0     493500.0  2.491678e+05
Ideal               100000.0     100000.0  1.000000e+05
Indian              700000.0    1900000.0  1.100000e+06
Jawa                146000.0     223000.0  1.855000e+05
KTM                  55000.0     860000.0  1.746697e+05
Kawasaki            110000.0    1100000.0  4.116246e+05
LML                   4400.0       4400.0  4.400000e+03
MV                  950000.0    1500000.0  1.325000e+06
Mahindra             17800.0     175000.0  7.250709e+04
Rajdoot              75000.0      75000.0  7.500000e+04
Royal Enfield        33500.0     285000.0  9.856207e+04
Suzuki                8000.0    1260000.0  4.594683e+04
TVS                   5800.0     224000.0  4.429915e+04
Triumph             500000.0    1300000.0  8.274230e+05
Yamaha                9400.0    1550000.0  5.706896e+04
Yezdi                68000.0      68000.0  6.800000e+04
```

2 MAP()

It IS USED IN ENCODING OR OTHER IN OTHER PROBLEMS WHERE HAVE TO ASSIGN MULTIPLE VALUES AT A TIME

```
[21]: v=list(df["owner"].value_counts().keys())
      v
```

```
[21]: ['First Owner', 'Second Owner', 'Third Owner', 'Fourth Owner Or More']
```

```
[27]: int_owner={'First Owner':1,
              'Second Owner':2,
              'Third Owner':3,
              'Fourth Owner Or More':4}
```

```
[28]: df["owner"]=df["owner"].map(int_owner)
```

```
[29]: df
```

```
[29]:                                   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          1   3.0  110.0            TVS
      1          1   4.0  350.0  Royal Enfield
      2          1   8.0  675.0        Triumph
      3          1   4.0  180.0            TVS
      4          1   3.0  150.0         Yamaha
      …        …    …     …             …
      32643      1   4.0  100.0           Hero
      32644      1   9.0  180.0            TVS
      32645      1   6.0  220.0          Bajaj
      32646      1  16.0  125.0           Hero
      32647      1  13.0  150.0          Bajaj

      [32648 rows x 8 columns]
```

3 Other important functions

.concat()

.merg()

apply()