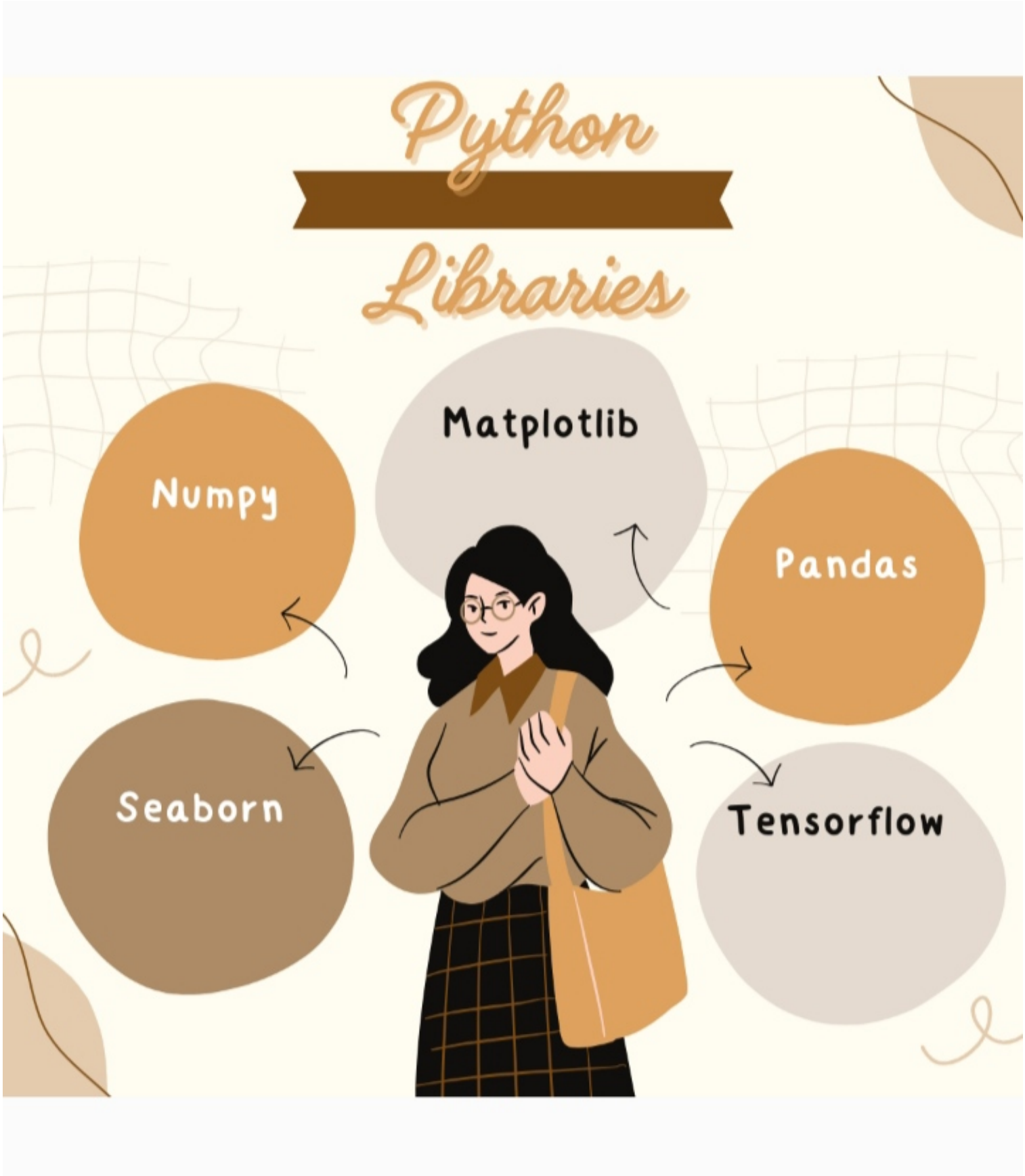


VISHNU VARDHAN

Pandas

is a powerful and flexible open-source data analysis and manipulation library for the Python programming language. It provides data structures and functions needed to work on structured data seamlessly and efficiently. Here are some key aspects of Pandas:

<————-Key Features————->

Data Structures: *Series*: A one-dimensional array-like object containing an array of data and an associated array of data labels (index). *DataFrame*: A two-dimensional, size-mutable, and potentially heterogeneous tabular data structure with labeled axes (rows and columns). Data Manipulation:

Data Cleaning: Handle missing data, data alignment, and reshaping.

Merging and Joining: Combine data from different sources.

Reshaping: Pivoting and stacking.

Slicing and Indexing: Access and modify data subsets.

Group By: Split-apply-combine operations on data.

Input/Output:Supports reading and writing to various file formats, such as CSV, Excel, SQL databases, and JSON.

Time Series: Powerful tools for working with time-series data, including date range generation, frequency conversion, moving window statistics, and more.

Visualization:Integration with Matplotlib to generate plots and visualizations directly from data structures. Performance:

Installation of Pandas

[127]:
```
! pip install pandas
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pandas in
c:\users\hp\appdata\roaming\python\python312\site-packages (2.2.2)
Requirement already satisfied: numpy>=1.26.0 in
c:\users\hp\appdata\roaming\python\python312\site-packages (from pandas) (2.0.1)
Requirement already satisfied: python-dateutil>=2.8.2 in
c:\users\hp\appdata\roaming\python\python312\site-packages (from pandas)
(2.9.0.post0)
```

```
Requirement already satisfied: pytz>=2020.1 in
c:\users\hp\appdata\roaming\python\python312\site-packages (from pandas)
(2024.1)
Requirement already satisfied: tzdata>=2022.7 in
c:\users\hp\appdata\roaming\python\python312\site-packages (from pandas)
(2024.1)
Requirement already satisfied: six>=1.5 in
c:\users\hp\appdata\roaming\python\python312\site-packages (from python-
dateutil>=2.8.2->pandas) (1.16.0)


[notice] A new release of pip is available: 24.0 -> 24.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Importing pandas module

[128]:
```python
import pandas as pd
import numpy as np
```

<——-DataFrame Creation——-> from_dict: Create a DataFrame from a dictionary of array-like or dicts.

[129]:
```python
data={"Name":["sachine","lakshay","saurabh","abhishek"],
      "Branch":["CSE","IT","AI","ECE"],
            "College":["Piet","Arya","Jcerc","Raffles"],
                  "Roll_no":[10,20,30,40] }
data
```

[129]:
```
{'Name': ['sachine', 'lakshay', 'saurabh', 'abhishek'],
 'Branch': ['CSE', 'IT', 'AI', 'ECE'],
 'College': ['Piet', 'Arya', 'Jcerc', 'Raffles'],
 'Roll_no': [10, 20, 30, 40]}
```

DataFrame: Two-dimensional, size-mutable, and potentially heterogeneous tabular data.

[130]:
```python
df=pd.DataFrame(data)
df
```

[130]:

	Name	Branch	College	Roll_no
0	sachine	CSE	Piet	10
1	lakshay	IT	Arya	20
2	saurabh	AI	Jcerc	30
3	abhishek	ECE	Raffles	40

Series: One-dimensional array-like structure.

[131]:
```python
df['Name']
s = pd.Series([1, 2, 3])
s
```

[131]:
```
0    1
1    2
2    3
dtype: int64
```

Type of data series:—

[132]:
```
type(df['Name'])
```

[132]:
```
pandas.core.series.Series
```

Multiple data series fetching:—-

[133]:
```
df[['Name','Branch','College']]
```

[133]:
```
       Name Branch  College
0   sachine    CSE     Piet
1   lakshay     IT     Arya
2   saurabh     AI    Jcerc
3  abhishek    ECE  Raffles
```

[134]:
```
df.astype
```

[134]:
```
<bound method NDFrame.astype of        Name Branch  College  Roll_no
0   sachine    CSE     Piet       10
1   lakshay     IT     Arya       20
2   saurabh     AI    Jcerc       30
3  abhishek    ECE  Raffles       40>
```

Basic Operations:——

head: Return the first n rows.

[135]:
```
df.head()
```

[135]:
```
       Name Branch  College  Roll_no
0   sachine    CSE     Piet       10
1   lakshay     IT     Arya       20
2   saurabh     AI    Jcerc       30
3  abhishek    ECE  Raffles       40
```

tail: Return the last n rows.

[136]:
```
df.tail()
```

[136]:
```
       Name Branch  College  Roll_no
0   sachine    CSE     Piet       10
1   lakshay     IT     Arya       20
2   saurabh     AI    Jcerc       30
```

```
 3  abhishek    ECE  Raffles       40
```

info: Print a concise summary of a DataFrame.

[137]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 4 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Name     4 non-null      object
 1   Branch   4 non-null      object
 2   College  4 non-null      object
 3   Roll_no  4 non-null      int64
dtypes: int64(1), object(3)
memory usage: 260.0+ bytes
```

describe: Generate descriptive statistics.

[138]:
```
df.describe()
```

[138]:
```
         Roll_no
count   4.000000
mean   25.000000
std    12.909944
min    10.000000
25%    17.500000
50%    25.000000
75%    32.500000
max    40.000000
```

shape: Return a tuple representing the dimensionality.

[139]:
```
df.shape
```

[139]:
```
(4, 4)
```

Selection and Filtering:—-

loc: Access a group of rows and columns by labels or a boolean array.

[140]:
```
df.loc[1:2][['Name','Branch','College']]
```

[140]:
```
      Name Branch College
1  lakshay     IT    Arya
2  saurabh     AI   Jcerc
```

iloc: Purely integer-location based indexing for selection by position.

[141]:
```
df.iloc[1:3][['Name','Branch','College']]
```

[141]:
```
      Name Branch College
1  lakshay     IT    Arya
2  saurabh     AI   Jcerc
```

at: Access a single value for a row/column label pair.

[142]:
```
df.at[0, 'Name']
```

[142]:
```
'sachine'
```

iat: Access a single value for a row/column pair by integer position.

[143]:
```
df.iat[0, 0]
```

[143]:
```
'sachine'
```

Data Manipulation——-

drop: Drop specified labels from rows or columns.

[144]:
```
df.drop(['Name'], axis=1)
```

[144]:
```
  Branch  College  Roll_no
0    CSE     Piet       10
1     IT     Arya       20
2     AI    Jcerc       30
3    ECE  Raffles       40
```

rename: Alter axes labels.

[145]:
```
df.rename(columns={'Name': 'F_Name'})
```

[145]:
```
     F_Name Branch  College  Roll_no
0   sachine    CSE     Piet       10
1   lakshay     IT     Arya       20
2   saurabh     AI    Jcerc       30
3  abhishek    ECE  Raffles       40
```

sort_values: Sort by the values along either axis.

[146]:
```
df.sort_values(by='Name')
```

[146]:
```
       Name Branch  College  Roll_no
3  abhishek    ECE  Raffles       40
1   lakshay     IT     Arya       20
0   sachine    CSE     Piet       10
2   saurabh     AI    Jcerc       30
```

sort_index: Sort by the index.

[147]:
```
df.sort_index()
```

[147]:
```
       Name Branch  College  Roll_no
0   sachine    CSE     Piet       10
1   lakshay     IT     Arya       20
2   saurabh     AI    Jcerc       30
3  abhishek    ECE  Raffles       40
```

groupby: Group DataFrame using a mapper or by a Series of columns.

[148]:
```
df.groupby('Name').sum()
```

[148]:
```
         Branch  College  Roll_no
Name
abhishek    ECE  Raffles       40
lakshay      IT     Arya       20
sachine     CSE     Piet       10
saurabh      AI    Jcerc       30
```

map: Map values of Series according to an input mapping.

[ ]:
```
df['Name'].map({1: 'sachine', 2: 'two', 3: 'three'})
```

[ ]:
```
0    NaN
1    NaN
2    NaN
3    NaN
Name: Name, dtype: object
```

[ ]:
```
data1={"Name":["Ayush","ISha","sam","rohit"],
      "Branch":["CSE","IT","AI","ECE"],
            "College":["Piet","PIet","Jcerc","Raffles"],
                  "Roll_no":[24,50,36,21] }
data
df1=pd.DataFrame(data1)
df1
```

[ ]:
```
    Name Branch  College  Roll_no
0  Ayush    CSE     Piet       24
1   ISha     IT     PIet       50
2    sam     AI    Jcerc       36
3  rohit    ECE  Raffles       21
```

merge: Merge DataFrame or named Series objects with a database-style join.

[ ]:
```
pd.merge(df1, df, on='College')
```

```
[ ]:   Name_x Branch_x  College  Roll_no_x    Name_y Branch_y  Roll_no_y
     0  Ayush      CSE     Piet         24   sachine      CSE         10
     1    sam       AI    Jcerc         36   saurabh       AI         30
     2  rohit      ECE  Raffles         21  abhishek      ECE         40
```

concat: Concatenate pandas objects along a particular axis.

```
[ ]: pd.concat([df1, df])
```

```
[ ]:        Name Branch  College  Roll_no
     0     Ayush    CSE     Piet       24
     1      ISha     IT     PIet       50
     2       sam     AI    Jcerc       36
     3     rohit    ECE  Raffles       21
     0   sachine    CSE     Piet       10
     1   lakshay     IT     Arya       20
     2   saurabh     AI    Jcerc       30
     3  abhishek    ECE  Raffles       40
```

pivot_table: Create a spreadsheet-style pivot table as a DataFrame.

```
[ ]: df.pivot_table(values='Roll_no', index=['Name', 'College'], columns=['Branch'])
```

```
[ ]: Branch              AI   CSE   ECE    IT
     Name     College
     abhishek Raffles   NaN   NaN  40.0   NaN
     lakshay  Arya      NaN   NaN   NaN  20.0
     sachine  Piet      NaN  10.0   NaN   NaN
     saurabh  Jcerc    30.0   NaN   NaN   NaN
```

Handling Missing Data

isna: Detect missing values.

```
[ ]: df.isna()
```

```
[ ]:     Name  Branch  College  Roll_no
     0  False   False    False    False
     1  False   False    False    False
     2  False   False    False    False
     3  False   False    False    False
```

fillna: Fill NA/NaN values.

```
[ ]: df.fillna(0)
```

```
[ ]:        Name Branch  College  Roll_no
     0   sachine    CSE     Piet       10
     1   lakshay     IT     Arya       20
```

```
2   saurabh     AI    Jcerc       30
3  abhishek    ECE  Raffles       40
```

dropna: Remove missing values.

```
[ ]: df.dropna()
```

```
[ ]:        Name Branch  College  Roll_no
0   sachine    CSE     Piet       10
1   lakshay     IT     Arya       20
2   saurabh     AI    Jcerc       30
3  abhishek    ECE  Raffles       40
```

interpolate: Interpolate values according to different methods.

```
[ ]: df.interpolate()
```

```
[ ]:    Age
0   25
1   30
2   35
```

Data Input/Output

to_csv: Write DataFrame to a CSV file.

```
[ ]: import pandas as pd
import numpy as np
file_path = r"C:\Users\HP\Downloads\\Used_Bikes.csv"
d= pd.read_csv(file_path)
# r is know as raw string
# first go to properties then go to security
d
```

```
[ ]:                                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]
```

[ ]:
```
d
```

[ ]:
```
                                 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]
```

To check which value has duplicate value in boolean form

[ ]:
```
d.duplicated()
```

[ ]:
```
0        False
1        False
2        False
3        False
4        False
         …
32643     True
32644     True
32645     True
32646     True
32647     True
Length: 32648, dtype: bool
```

To count the duplicate values

[ ]:
```
d.duplicated().sum()
```

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

To make the changes permanent to the data by putting (inplace=True)

[ ]:
```
d.drop_duplicates(inplace=True)
```

To count the number of unique brands present

[ ]:
```
d['brand'].nunique()
```

[ ]:
```
23
```

To give the names of all brand without repeating anyone

[ ]:
```
d['brand'].unique()
```

[ ]:
```
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)
```

To count the numbers of item present in each brand

[ ]:
```
d['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
```

Filtering the Data according to use:–

[ ]:
```
d['brand']
```

[ ]:
```
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
```

Filtering the brand data whose brand is Royal Enfield

[ ]:
```
d[d['brand']=="Royal Enfield"]
```

[ ]:
```
                                    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]
```

[ ]:
```
bullet=d[d['brand']=="TVS"]
bullet
```

[ ]:
```
                              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
…                                    …      …         …          …
9247      TVS Apache RTR 160cc Rear Disc  70000.0  Ghaziabad      4116.0
9307                TVS Apache RTR 160cc  30000.0     Alibag     30000.0
9312            TVS Apache RTR 200 4V FI  65450.0      Delhi      9238.0
9320                    TVS Apache 150cc  20000.0     Hissar     84916.0
9322           TVS Radeon 110cc Drum SBT  58000.0      Delhi      4020.0

            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
…           …   …    …   …
9247  First Owner   3.0  160.0   TVS
9307  First Owner  10.0  160.0   TVS
9312  First Owner   3.0  200.0   TVS
9320  First Owner  14.0  150.0   TVS
```

```
9322  First Owner   2.0  110.0   TVS

[481 rows x 8 columns]
```

Filtering the data of owner which is Third Owner

```
[ ]: d[d['owner']=="Third Owner"]
```

```
[ ]:                                    bike_name     price        city  \
103                        Bajaj Pulsar 150cc   11100.0        Pune
225                              KTM RC 200cc  113000.0    Kottayam
284            Triumph Street Triple ABS 675cc  599999.0      Mumbai
364                       Kawasaki Ninja 250cc  125000.0  Coimbatore
474            Mahindra Centuro Rockstar 110cc   18999.0        Pune
…                                           …       …         …
8676  Royal Enfield Classic Desert Storm 500cc   92000.0    Ludhiana
8791                         Bajaj Pulsar 220F   21305.0   Ahmedabad
8876                         Yamaha FZ16 150cc   65000.0      Rajkot
9183           Royal Enfield Thunderbird 350cc   35800.0   Bangalore
9350                        Bajaj Pulsar NS200   46000.0   Bangalore

      kms_driven        owner   age  power          brand
103      12000.0  Third Owner  12.0  150.0          Bajaj
225      28000.0  Third Owner   6.0  200.0            KTM
284       7800.0  Third Owner   5.0  675.0        Triumph
364      21000.0  Third Owner  11.0  250.0       Kawasaki
474      38669.0  Third Owner   6.0  110.0       Mahindra
…          …          …   …    …            …
8676     23569.0  Third Owner   7.0  500.0  Royal Enfield
8791     60000.0  Third Owner  11.0  220.0          Bajaj
8876     18000.0  Third Owner   8.0  150.0         Yamaha
9183     90408.0  Third Owner  18.0  350.0  Royal Enfield
9350     27687.0  Third Owner   9.0  200.0          Bajaj

[84 rows x 8 columns]
```

```
[ ]: bullet2=d[(d['brand']=="Royal Enfield") & (d['age']<=4) & (d['owner']=="First␣
     ↪Owner")]
     bullet2
```

```
[ ]:                                    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]
```

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

[ ]:
```
                         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
```

[ ]:
```
brands =['Bajaj','TVS','Hero','Yamaha']
d[d['brand'].isin(brands)]
```

[ ]:
```
                                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]
```

Drop function is used to delete the row or column according to axis which choose axis:1—->for column axis:0—->for row

[ ]:
```
# to remove particular col
d.drop(['bike_name','owner'],axis='columns',inplace=True)
```

[ ]:
```
d.drop('price',axis=1)
# axis-->row,column
# row--->0
# col-->1
```

[ ]:
```
           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]
```

Adding a new column named “B” and its values are upflairs

[ ]:
```
d['B']="upflairs"
# scalar values
```

[ ]:
```
d['price']+5000
```

[ ]:
```
0        40000.0
1       124900.0
2       605000.0
3        70000.0
4        85000.0
          …
9362     30000.0
9369     40000.0
9370    455000.0
9371    144000.0
9372     85000.0
Name: price, Length: 7324, dtype: float64
```

This feature called "feature engineering" is used here which means new columne is build using the existing column

[ ]:
```
d['Updated_price']=d['price']+5000
# feature engineering:new column building using existing col
```

#Data cleaning –>missing value handling

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

[ ]:
```
data={'A':[10,np.nan,20,54,30,40,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.nan=null value stored
```

[ ]:
```
df2=pd.DataFrame(data)
df2
```

[ ]:

	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	54.0	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

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

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

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

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

Fill the missing value or remove the missing value——————-> (1)row Wise (2)Column wise

[ ]:
```
df2.dropna()
# to drop the null values
# for permanent change --->inplace=True
```

[ ]:
```
      A     B     c  D
3  54.0  70.0  70.0  4
5  40.0  25.0  90.0  6
```

[ ]:
```
# df2.dropna(axis='columns')      or
df2.dropna(axis=1)
```

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

Filling the records which has null value filling the scaler or constant value

[ ]:
```
df2.fillna("Upflairs")
```

[ ]:
```
          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      54.0      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
```

[ ]:
```
df2['B'].fillna("Upflairs")
# inplace=True
```

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

before filling null value(na) column–>numerial —->constant,mean or median or categorical —->constant,mode

[ ]:
```
df2['c'].mean()
```

[ ]:
```
np.float64(76.6)
```

[ ]:
```
df2['c'].fillna(76.6)
```

[ ]:
```
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
```

[ ]:
```
df
```

[ ]:
```
   Age
0   25
1   30
2   35
```

[ ]:
```
t=d[(d['brand']=="Royal Enfield") ]
t['price'].min()
```

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

[ ]:
```
# how to find the minimum price from all the brand
min_price = d['price'].min()
print(f"The minimum price from all bikes is: {min_price}")
```

```
The minimum price from all bikes is: 4400.0
```

[ ]:
```
# how to find the minimum price from each the brand
min_price_by_brand = d.groupby('brand')[['price']].min()
print(min_price_by_brand)
```

```
                     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
```

to_excel: Write DataFrame to an Excel file

[ ]:
```
df.to_excel('Used_Bikes.xlsx')
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]
```

Statistical Functions

mean: Return the mean of the values.

[ ]:
```
data = {
    'Age': [25, 30, 35],

}

df = pd.DataFrame(data)
df.mean()
```

[ ]:
```
Age    30.0
dtype: float64
```

sum: Return the sum of the values.

[ ]:
```
data = {
    'Age': [25, 30, 35],

}

df = pd.DataFrame(data)
df.sum()
```

[ ]:
```
Age    90
dtype: int64
```

std: Return sample standard deviation over requested axis.

[ ]:
```
data = {
    'Age': [25, 30, 35],

}

df = pd.DataFrame(data)
df.std()
```

[ ]:
```
Age    5.0
dtype: float64
```

min: Return the minimum of the values.

[ ]:
```
data = {
    'Age': [25, 30, 35],

}

df = pd.DataFrame(data)
df.min()
```

[ ]:
```
Age    25
dtype: int64
```

max: Return the maximum of the values.

[ ]:
```
data = {
    'Age': [25, 30, 35],

}

df = pd.DataFrame(data)
df.max()
```

[ ]:
```
Age    35
dtype: int64
```

count: Return the number of non-NA/null observations.

[ ]:
```
data = {
    'Age': [25, 30, 35],

}

df = pd.DataFrame(data)
df.count()
```

[ ]:
```
Age    3
dtype: int64
```

cumsum: Return cumulative sum over a DataFrame or Series axis.

[ ]:
```
data = {
    'Age': [25, 30, 35],

}
```

```
df = pd.DataFrame(data)
df.cumsum()
```

[ ]:

```
   Age
0   25
1   55
2   90
```

cumprod: Return cumulative product over a DataFrame or Series axis.

[ ]:

```
data = {
    'Age': [25, 30, 35],

}

df = pd.DataFrame(data)
df.cumprod()
```

[ ]:

```
     Age
0     25
1    750
2  26250
```

Visualization

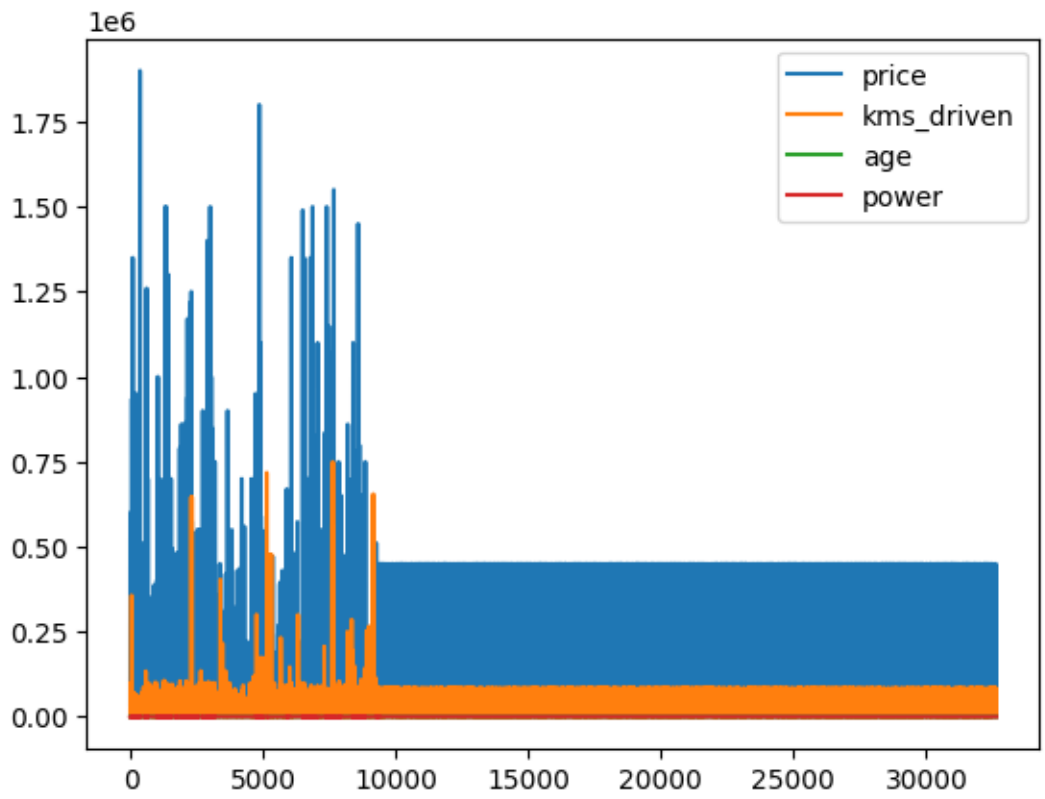
plot: Make plots of Series or DataFrame.

[ ]:

```
df.plot()
```

[ ]:

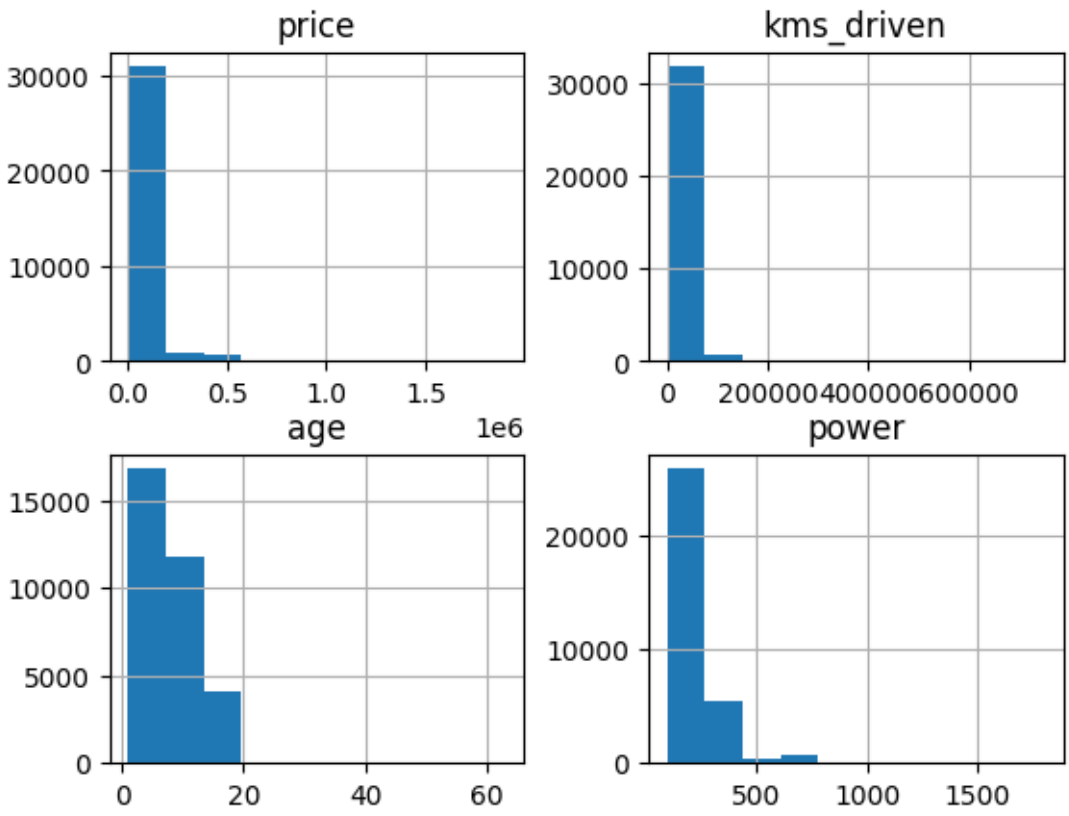
```
<Axes: >
```



hist: Make a histogram of the DataFrame's columns.

[ ]:
```
df.hist()
```

[ ]:
```
array([[<Axes: title={'center': 'price'}>,
        <Axes: title={'center': 'kms_driven'}>],
       [<Axes: title={'center': 'age'}>,
        <Axes: title={'center': 'power'}>]], dtype=object)
```


price
30000
20000
10000
0
0.0
0.5
1.0
1.5
kms_driven
30000
20000
10000
0
0
200000400000600000
age
1e6
15000
10000
5000
0
0
20
40
60
power
20000
10000
0
500
1000
1500