What is Pandas?

Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

IMPORT Libraries of python

import pandas as pd import numpy as np

In [1]:

```
import pandas as pd
import numpy as np
```

In [2]:

```
# Check version of pandas
pd.__version__
```

Out[2]:

```
'2.2.2'
```

What is dataframe? How can we create dataframe, array, list, dictionary ?

In [3]:

```
# Let take a dictionary

dt = {"Name":['Ritu','Taniya','abhishek','tanmay','mayank'],
      "Branch":['it','cse','it','ece','cse'],
      "Class":[2,3,4,1,2],
      "City":['jaipur','agra','vridavan','mathura','gokul']}
dt
```

Out[3]:

```
{'Name': ['Ritu', 'Taniya', 'abhishek', 'tanmay', 'mayank'],
 'Branch': ['it', 'cse', 'it', 'ece', 'cse'],
 'Class': [2, 3, 4, 1, 2],
 'City': ['jaipur', 'agra', 'vridavan', 'mathura', 'gokul']}
```

DATAFRAME-------- A dataframe is a data structure constructed with rows and columns, similar to a database or Excel spreadsheet. It consists of a dictionary of lists in which the list each have their own identifiers or keys, such as “last name” or “food group.”

In [4]:

```
#np.array()
df = pd.DataFrame(dt)
df
```

Out[4]:

	Name	Branch	Class	City
0	Ritu	it	2	jaipur
1	Taniya	cse	3	agra
2	abhishek	it	4	vridavan
3	tanmay	ece	1	mathura
4	mayank	cse	2	gokul

In [5]:

```
# check the type of dataFrame
type(df)
```

Out[5]:

```
pandas.core.frame.DataFrame
```

SERIES--> show a single col

In [6]:

```
df['Name']
```

Out[6]:

```
0        Ritu
1      Taniya
2    abhishek
3      tanmay
4      mayank
Name: Name, dtype: object
```

In [7]:

```
# type of name column-----> Series
type(df['Name'])
```

Out[7]:

```
pandas.core.series.Series
```

In [8]:

```
# String,categorical data values represent as Object

df.dtypes
```

Out[8]:

```
Name      object
Branch    object
Class      int64
City      object
dtype: object
```

Shape--> Define the number of rows and columns

In [9]:

```
#5-> row, 4-> col
df.shape
```

Out[9]:

```
(5, 4)
```

To fetch a single col

In [10]:

```
# df['Name']
# OR
df.Name
```

Out[10]:

```
0        Ritu
1      Taniya
2    abhishek
3      tanmay
```

```
4      mayank
Name: Name, dtype: object
```

In [11]:

```
# To represent a "Branch" & "Name" Column from table
df[['Branch','Name']]
```

Out[11]:

	Branch	Name
0	it	Ritu
1	cse	Taniya
2	it	abhishek
3	ece	tanmay
4	cse	mayank

In [12]:

```
# It is used to show the DataFrame/Table
df
```

Out[12]:

	Name	Branch	Class	City
0	Ritu	it	2	jaipur
1	Taniya	cse	3	agra
2	abhishek	it	4	vridavan
3	tanmay	ece	1	mathura
4	mayank	cse	2	gokul

In [13]:

```
# Select only the 'Branch' and 'Class' columns
df[['Branch','Class']]
```

Out[13]:

	Branch	Class
0	it	2
1	cse	3
2	it	4
3	ece	1
4	cse	2

To fetch a row ---> Use these methods .loc[]: Used for label-based indexing. It accepts a single label or a list of labels. .iloc[]: Used for positional indexing. It accepts integers or boolean arrays.

In [14]:

```
# Select rows from index 1 to 2 (3 is exclusive)
df[1:3]
```

Out[14]:

	Name	Branch	Class	City
1	Taniya	cse	3	agra
2	abhishek	it	4	vridavan

In [15]:

```
# It showed only 1st row
df[1:2]
```

Out[15]:

	Name	Branch	Class	City
1	Taniya	cse	3	agra

In [16]:

```
df[1:3][['Name','Branch']]
```

Out[16]:

	Name	Branch
1	Taniya	cse
2	abhishek	it

loc $ iloc

In [17]:

```
#loc-> stopping pnt is inclusive
df.loc[1:2,['Name','Branch']]
```

Out[17]:

	Name	Branch
1	Taniya	cse
2	abhishek	it

In [18]:

```
#iloc-> stoping pnt is not inclusive
df.iloc[1:3,0:2]
```

Out[18]:

	Name	Branch
1	Taniya	cse
2	abhishek	it

In [19]:

```
#Selected Subset (Rows 3 onwards, Columns 1 onwards):
df.iloc[3:,1:]
```

Out[19]:

	Branch	Class	City
3	ece	1	mathura
4	cse	2	gokul

In [20]:

```
# Select rows with labels 3 to 4 and columns 'Branch', 'Class', 'City'
df.loc[3:4,['Branch','Class','City']]
df.iloc[3:,1:]
```

Out[20]:

	Branch	Class	City
3	ece	1	mathura
4	cse	2	gokul

Use read_csv(): The pd.read_csv() function is used to read the CSV file into a pandas DataFrame. You need to provide the path to your CSV file within the function call. If your CSV file is in a different directory, specify the full path or relative path accordingly.

In [22]:

```
df = pd.read_csv('Used_Bikes.csv')
df
```

Out[22]:

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

32648 rows × 8 columns

Displaying the DataFrame: Optionally, you can print the first few rows of the DataFrame using .head() to verify that the CSV file was read correctly.

In [23]:

```
# Display the first few rows of the DataFrame
df.head()
```

Out[23]:

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

In [24]:

```
# Display the first 10 rows of the DataFrame
df.head(10)
```

Out[24]:

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

Displaying the DataFrame: Optionally, you can print the last few rows of the DataFrame using .tail() to verify that the CSV file was read correctly.

In [25]:

```
# Display the last 10 rows of the DataFrame
df.tail(10)
```

Out[25]:

	bike_name	price	city	kms_driven	owner	age	power	brand
32638	Yamaha Fazer 25 250cc	123000.0	Kadapa	14500.0	First Owner	4.0	250.0	Yamaha
32639	Royal Enfield Classic 350cc	95500.0	Delhi	18000.0	First Owner	8.0	350.0	Royal Enfield
32640	Hero Passion Pro 100cc	32000.0	Delhi	12000.0	First Owner	6.0	100.0	Hero
32641	Bajaj Avenger 220cc	41000.0	Delhi	20245.0	Second Owner	11.0	220.0	Bajaj
32642	Hero Passion 100cc	15000.0	Perumbavoor	35000.0	Second Owner	19.0	100.0	Hero
32643	Hero Passion Pro 100cc	39000.0	Delhi	22000.0	First Owner	4.0	100.0	Hero
32644	TVS Apache RTR 180cc	30000.0	Karnal	6639.0	First Owner	9.0	180.0	TVS
32645	Bajaj Avenger Street 220	60000.0	Delhi	20373.0	First Owner	6.0	220.0	Bajaj
32646	Hero Super Splendor 125cc	15600.0	Jaipur	84186.0	First Owner	16.0	125.0	Hero
32647	Bajaj Pulsar 150cc	22000.0	Pune	60857.0	First Owner	13.0	150.0	Bajaj

In [26]:

```
# Access the 'brand' column
df['brand']
```

Out[26]:

```
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
```

no. of unique class

In [27]:

```
#The .nunique() method in pandas is used to count the number of unique elements in a Series or DataFrame column.
df['brand'].nunique()
```

Out[27]:

```
23
```

total unique name of class

In [28]:

```
#the unique() function is used to extract the unique elements from a Series. It returns a NumPy array containing only the unique values present in the Series.
df['brand'].unique()
```

Out[28]:

```
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)
```

How many bikes available of individual brand ?

In [29]:

```
#value_counts() is a powerful method used to count the frequency of unique values in a Series. It returns a Series containing counts of unique values. This method is particularly useful for categorical data analysis and data exploration tasks.
df['brand'].value_counts()
```

Out[29]:

```
brand
Bajaj               11213
Hero                 6368
Royal Enfield        4178
Yamaha               3916
Honda                2108
Suzuki               1464
TVS                  1247
KTM                  1077
Harley-Davidson       737
Kawasaki               79
Hyosung                64
Benelli                56
Mahindra               55
Triumph                26
Ducati                 22
BMW                    16
Jawa                   10
MV                      4
Indian                  3
Ideal                   2
Rajdoot                 1
Yezdi                   1
LML                     1
Name: count, dtype: int64
```

In [30]:

```
# Example: Filter rows where 'brand' column equals 'Royal Enfield'
bullete = df[df['brand'] == "Royal Enfield"] # single cndn
```

In [31]:

```
#bullete.head() will output the first 5 rows (by default) of the DataFrame bullete, which contains only the rows where 'brand' is 'Royal Enfield'.
bullete.head()
```

Out[31]:

	bike_name	price	city	kms_driven	owner	age	power	brand
1	Royal Enfield Classic 350cc	119900.0	Delhi	11000.0	First Owner	4.0	350.0	Royal Enfield
8	Royal Enfield Thunderbird X 350cc	145000.0	Bangalore	9190.0	First Owner	3.0	350.0	Royal Enfield
9	Royal Enfield Classic Desert Storm 500cc	88000.0	Delhi	19000.0	Second Owner	7.0	500.0	Royal Enfield
23	Royal Enfield Classic Chrome 500cc	121700.0	Kalyan	24520.0	First Owner	5.0	500.0	Royal Enfield
36	Royal Enfield Classic 350cc	98800.0	Kochi	39000.0	First Owner	5.0	350.0	Royal Enfield

In [32]:

```
#shape gives you the number of rows and columns.
bullete.shape
```

Out[32]:

```
(4178, 8)
```

In [33]:

```
#brand = royal enf
#age== less thn 2 yr
# owner== first owner

df[(df['brand']=="Royal Enfield") & (df['age']<=2) & (df['owner']=="First Owner")]
```

Out[33]:

	bike_name	price	city	kms_driven	owner	age	power	brand
38	Royal Enfield Thunderbird X 500cc	190500.0	Samastipur	4550.0	First Owner	2.0	500.0	Royal Enfield
81	Royal Enfield Interceptor 650cc	260000.0	Navi Mumbai	3800.0	First Owner	2.0	650.0	Royal Enfield
139	Royal Enfield Himalayan 410cc Fi ABS	173300.0	Vadodara	14000.0	First Owner	2.0	410.0	Royal Enfield
157	Royal Enfield Himalayan 410cc Fi ABS	173300.0	Vadodara	14000.0	First Owner	2.0	410.0	Royal Enfield
194	Royal Enfield Electra 350cc	145000.0	Bangalore	4000.0	First Owner	2.0	350.0	Royal Enfield
...	...	...	...	...	...	...	...	...
7694	Royal Enfield Classic Chrome 500cc ABS	215000.0	Delhi	417.0	First Owner	2.0	500.0	Royal Enfield
7706	Royal Enfield Classic Chrome 500cc ABS	215000.0	Delhi	417.0	First Owner	2.0	500.0	Royal Enfield
8139	Royal Enfield Thunderbird X 350cc ABS	169000.0	Bangalore	4411.0	First Owner	2.0	350.0	Royal Enfield
8192	Royal Enfield Thunderbird 350cc ABS	145000.0	Ghaziabad	12400.0	First Owner	2.0	350.0	Royal Enfield
8836	Royal Enfield Thunderbird X 350cc ABS	170200.0	Mumbai	1000.0	First Owner	2.0	350.0	Royal Enfield

84 rows × 8 columns

In [34]:

```
# Apply the filter to get specific rows
bullete = df[(df['brand']=="Royal Enfield") & (df['age']<=2) & (df['owner']=="First Own
er")]
```

In [35]:

```
bullete.head()
```

Out[35]:

	bike_name	price	city	kms_driven	owner	age	power	brand
38	Royal Enfield Thunderbird X 500cc	190500.0	Samastipur	4550.0	First Owner	2.0	500.0	Royal Enfield
81	Royal Enfield Interceptor 650cc	260000.0	Navi Mumbai	3800.0	First Owner	2.0	650.0	Royal Enfield

	bike_name	price	city	kms_driven	owner	age	power	brand
139	Royal Enfield Himalayan 410cc Fi ABS	173300.0	Vadodara	14000.0	First Owner	2.0	410.0	Royal Enfield
157	Royal Enfield Himalayan 410cc Fi ABS	173300.0	Vadodara	14000.0	First Owner	2.0	410.0	Royal Enfield
194	Royal Enfield Electra 350cc	145000.0	Bangalore	4000.0	First Owner	2.0	350.0	Royal Enfield

In [36]:

```
#shape returns a tuple (number_of_rows, number_of_columns)
bullete.shape
```

Out[36]:

```
(84, 8)
```

In [37]:

```
# Find the number of unique cities in the 'city' column
bullete['city'].nunique()
```

Out[37]:

```
28
```

In [38]:

```
# Find the unique cities in the 'city' column
bullete['city'].unique()
```

Out[38]:

```
array(['Samastipur', 'Navi Mumbai', 'Vadodara', 'Bangalore',
       'Hamirpur(hp)', 'Mumbai', 'Delhi', 'Guwahati', 'Haldwani',
       'Ahmedabad', 'Bardhaman', 'Silchar', 'Sibsagar', 'Kharar',
       'Baripara', 'Sonipat', 'Pune', 'Farukhabad', 'Sultanpur',
       'Hyderabad', 'Gurgaon', 'Faridabad', 'Kota', 'Thane', 'Nellore',
       'Alipore', 'Ghaziabad', 'Noida'], dtype=object)
```

In [39]:

```
# Get the count of each unique city in the 'city' column
bullete['city'].value_counts()
```

Out[39]:

```
city
Delhi           18
Mumbai          10
Bangalore        6
Ahmedabad        6
Vadodara         4
Guwahati         4
Faridabad        4
Hyderabad        3
Ghaziabad        3
Sibsagar         2
Bardhaman        2
Navi Mumbai      2
Gurgaon          2
Alipore          2
Farukhabad       2
Kharar           2
Silchar          1
Haldwani         1
Hamirpur(hp)     1
Samastipur       1
Sultanpur        1
Pune             1
Baripara         1
Sonipat          1
Thane            1
Kota             1
Nellore          1
Noida            1
```

```
Name: count, dtype: int64
```

In [40]:

```
bullete = df[(df['city']=="Jaipur") & (df['age']<=2) & (df['owner']=="First Owner")]
```

In [41]:

```
bullete.shape
```

Out[41]:

```
(13, 8)
```

In [42]:

```
df
```

Out[42]:

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

32648 rows × 8 columns

In [43]:

```
df['brand'].unique()
```

Out[43]:

```
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)
```

In [44]:

```
#The "|" operator performs element-wise OR operation between the two boolean Series. Thi
s results in a combined boolean Series where each element is True if the corresponding r
ow satisfies either condition
bike = df[(df['brand'] == "KTM") | (df['brand'] == "Jawa")]
bike
```

Out[44]:

	bike_name	price	city	kms_driven	owner	age	power	brand
33	KTM RC 390cc	180000.0	Pune	17700.0	First Owner	4.0	390.0	KTM
35	KTM Duke 200cc	70000.0	Nashik	100000.0	Second Owner	8.0	200.0	KTM
39	KTM RC 200cc ABS	179000.0	Bangalore	3400.0	First Owner	2.0	200.0	KTM
65	KTM Duke 200cc	94700.0	Baripara	32700.0	First Owner	4.0	200.0	KTM

	bike_name	price	city	kms_driven	owner	age	power	brand
83	KTM Duke 250cc	130000.0	Gandhidham	17500.0	Second Owner	4.0	250.0	KTM
...	...	...	...	...	...	...	...	...
32541	KTM RC 390cc	196700.0	Mumbai	13216.0	First Owner	4.0	390.0	KTM
32560	KTM RC 390cc	196700.0	Mumbai	13216.0	First Owner	4.0	390.0	KTM
32579	KTM RC 390cc	196700.0	Mumbai	13216.0	First Owner	4.0	390.0	KTM
32598	KTM RC 390cc	196700.0	Mumbai	13216.0	First Owner	4.0	390.0	KTM
32636	KTM RC 390cc	196700.0	Mumbai	13216.0	First Owner	4.0	390.0	KTM

1087 rows × 8 columns

In [45]:

```
# Get the count of each unique brand in the 'brand' column
bike['brand'].value_counts()
```

Out[45]:

```
brand
KTM     1077
Jawa      10
Name: count, dtype: int64
```

Conditional Check:

In [46]:

```
if "Jaipur" in bike['city'].unique():
    print("present")
else:
    print("not present")
```

```
present
```

In [47]:

```
bike['city'].value_counts()
```

Out[47]:

```
city
Mumbai          703
Delhi            65
Bangalore        55
Pune             36
Hyderabad        27
               ...
Challakere        1
Hooghly           1
Bhubaneswar       1
Raigarh(mh)       1
Berhampore        1
Name: count, Length: 91, dtype: int64
```

city== jaipur

In [48]:

```
#bike[bike['city'] == "Jaipur" ]
```

In [49]:

```
#population data --> df
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32648 entries, 0 to 32647
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
```

```
 #   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
```

In [50]:

```
df.shape  #actual
```

Out[50]:

```
(32648, 8)
```

df.duplicated().sum()-----> In pandas is used to count the number of duplicate rows in a DataFrame df

In [51]:

```
df.duplicated().sum() # duplicates
```

Out[51]:

```
np.int64(25324)
```

df.drop_duplicates() method in pandas is used to remove duplicate rows from a DataFrame df.

In [52]:

```
df.drop_duplicates()
```

Out[52]:

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

7324 rows × 8 columns

NOW, THE changes are permanent after using inplace func.

In [53]:

```
# Remove duplicate rows
df.drop_duplicates(inplace=True)
```

In [54]:

```
df.shape
```

Out[54]:

```
(7324, 8)
```

In [55]:

```
32648 - 25324
```

Out[55]:

```
7324
```

In [56]:

```
bullet = df[df['brand'] == "Royal Enfield"]
```

In [57]:

```
bullet.shape
```

Out[57]:

```
(1346, 8)
```

In [58]:

```
#.sort_values() is a versatile method in pandas for rearranging data based on specified criteria, offering flexibility in handling various sorting scenarios within DataFrames.
bullet.sort_values(by="price")
```

Out[58]:

	bike_name	price	city	kms_driven	owner	age	power	brand
5811	Royal Enfield Thunderbird 350cc	33500.0	Delhi	49463.0	First Owner	16.0	350.0	Royal Enfield
7038	Royal Enfield Bullet Electra 350cc	35000.0	Delhi	60000.0	Fourth Owner Or More	18.0	350.0	Royal Enfield
9183	Royal Enfield Thunderbird 350cc	35800.0	Bangalore	90408.0	Third Owner	18.0	350.0	Royal Enfield
4664	Royal Enfield Bullet Electra 350cc	41000.0	Noida	120000.0	First Owner	17.0	350.0	Royal Enfield
2371	Royal Enfield Bullet 350 cc	45000.0	Gurgaon	40000.0	Second Owner	20.0	350.0	Royal Enfield
...	...	...	...	...	...	...	...	...
5912	Royal Enfield Interceptor 650cc	265500.0	Nellore	12000.0	First Owner	2.0	650.0	Royal Enfield
1931	Royal Enfield Interceptor 650cc	270000.0	Ahmedabad	6500.0	First Owner	2.0	650.0	Royal Enfield
277	Royal Enfield Interceptor 650cc	280000.0	Bangalore	1500.0	First Owner	2.0	650.0	Royal Enfield
2228	Royal Enfield Interceptor 650cc	280000.0	Mumbai	5000.0	First Owner	2.0	650.0	Royal Enfield
4912	Royal Enfield Continental GT 650cc	285000.0	Hyderabad	4500.0	First Owner	2.0	650.0	Royal Enfield

1346 rows × 8 columns

In [59]:

```
bullet.sort_values(by="price").head(10)
```

Out[59]:

	bike_name	price	city	kms_driven	owner	age	power	brand

	bike_name	price	city	kms_driven	owner	age	power	brand
5811	Royal Enfield Thunderbird 350cc	33500.0	Delhi	49463.0	First Owner	16.0	350.0	Royal Enfield
7038	Royal Enfield Bullet Electra 350cc	35000.0	Delhi	60000.0	Fourth Owner Or More	18.0	350.0	Royal Enfield
9183	Royal Enfield Thunderbird 350cc	35800.0	Bangalore	90408.0	Third Owner	18.0	350.0	Royal Enfield
4664	Royal Enfield Bullet Electra 350cc	41000.0	Noida	120000.0	First Owner	17.0	350.0	Royal Enfield
2371	Royal Enfield Bullet 350 cc	45000.0	Gurgaon	40000.0	Second Owner	20.0	350.0	Royal Enfield
5918	Royal Enfield Thunderbird 350cc	45000.0	Bangalore	93108.0	Third Owner	18.0	350.0	Royal Enfield
8475	Royal Enfield Thunderbird 350cc	45000.0	Delhi	45710.0	First Owner	16.0	350.0	Royal Enfield
6489	Royal Enfield Thunderbird 350cc	45918.0	Bangalore	51396.0	Second Owner	12.0	350.0	Royal Enfield
385	Royal Enfield Thunderbird 350cc	46000.0	Chennai	35000.0	First Owner	16.0	350.0	Royal Enfield
1227	Royal Enfield Thunderbird 350cc	47000.0	Faridabad	21000.0	First Owner	9.0	350.0	Royal Enfield

In [60]:

```
# Setting ascending=False means you want to sort the DataFrame in descending order of 'p
rice'
bullet.sort_values(by="price",ascending=False)
```

Out[60]:

	bike_name	price	city	kms_driven	owner	age	power	brand
4912	Royal Enfield Continental GT 650cc	285000.0	Hyderabad	4500.0	First Owner	2.0	650.0	Royal Enfield
277	Royal Enfield Interceptor 650cc	280000.0	Bangalore	1500.0	First Owner	2.0	650.0	Royal Enfield
2228	Royal Enfield Interceptor 650cc	280000.0	Mumbai	5000.0	First Owner	2.0	650.0	Royal Enfield
1931	Royal Enfield Interceptor 650cc	270000.0	Ahmedabad	6500.0	First Owner	2.0	650.0	Royal Enfield
5912	Royal Enfield Interceptor 650cc	265500.0	Nellore	12000.0	First Owner	2.0	650.0	Royal Enfield
...	...	...	...	...	...	...	...	...
2371	Royal Enfield Bullet 350 cc	45000.0	Gurgaon	40000.0	Second Owner	20.0	350.0	Royal Enfield
4664	Royal Enfield Bullet Electra 350cc	41000.0	Noida	120000.0	First Owner	17.0	350.0	Royal Enfield
9183	Royal Enfield Thunderbird 350cc	35800.0	Bangalore	90408.0	Third Owner	18.0	350.0	Royal Enfield
7038	Royal Enfield Bullet Electra 350cc	35000.0	Delhi	60000.0	Fourth Owner Or More	18.0	350.0	Royal Enfield
5811	Royal Enfield Thunderbird 350cc	33500.0	Delhi	49463.0	First Owner	16.0	350.0	Royal Enfield

1346 rows × 8 columns

In [61]:

```
# Sort by 'price' in descending order and get the top 10 rows
bullet.sort_values(by="price",ascending=False).head(10)
```

Out[61]:

	bike_name	price	city	kms_driven	owner	age	power	brand
4912	Royal Enfield Continental GT 650cc	285000.0	Hyderabad	4500.0	First Owner	2.0	650.0	Royal Enfield
277	Royal Enfield Interceptor 650cc	280000.0	Bangalore	1500.0	First Owner	2.0	650.0	Royal Enfield
2228	Royal Enfield Interceptor 650cc	280000.0	Mumbai	5000.0	First Owner	2.0	650.0	Royal Enfield
1931	Royal Enfield Interceptor 650cc	270000.0	Ahmedabad	6500.0	First Owner	2.0	650.0	Royal Enfield

	bike_name	price	city	kms_driven	owner	age	power	brand
5912	Royal Enfield Interceptor 650cc	265500.0	Nellore	12000.0	First Owner	2.0	650.0	Royal Enfield
1396	Royal Enfield Interceptor 650cc	265000.0	Delhi	11000.0	First Owner	2.0	650.0	Royal Enfield
428	Royal Enfield Interceptor 650cc	265000.0	Bangalore	12900.0	First Owner	2.0	650.0	Royal Enfield
2332	Royal Enfield Interceptor 650cc	265000.0	Delhi	8500.0	First Owner	2.0	650.0	Royal Enfield
81	Royal Enfield Interceptor 650cc	260000.0	Navi Mumbai	3800.0	First Owner	2.0	650.0	Royal Enfield
1251	Royal Enfield Interceptor 650cc	250000.0	Silchar	920.0	First Owner	2.0	650.0	Royal Enfield

find out or filter all the yamaha brand bike top 10 lowest kms

In [62]:

```
df.brand.value_counts()
```

Out[62]:

```
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
```

In [63]:

```
bullet = df[df['brand'] == "Yamaha"]
```

In [64]:

```
bullet.sort_values(by="kms_driven").head(10)
```

Out[64]:

	bike_name	price	city	kms_driven	owner	age	power	brand
628	Yamaha FZs 150cc	77500.0	Delhi	45.0	First Owner	3.0	150.0	Yamaha
2264	Yamaha YZF-R1M 1000cc	600000.0	Faridabad	152.0	First Owner	5.0	1000.0	Yamaha
2037	Yamaha YZF-R1M 1000cc	600000.0	Surat	183.0	First Owner	5.0	1000.0	Yamaha
8693	Yamaha FZ25 ABS 250cc	142000.0	Delhi	233.0	First Owner	2.0	250.0	Yamaha
4765	Yamaha YZF-R1 1000cc	850000.0	Guwahati	285.0	First Owner	3.0	1000.0	Yamaha
4200	Yamaha YZF-R1 1000cc	700000.0	Delhi	285.0	First Owner	3.0	1000.0	Yamaha
123	Yamaha YZF R6 600cc	500000.0	Bangalore	285.0	First Owner	5.0	600.0	Yamaha
2265	Yamaha RX135 135cc 4-Speed	85000.0	Mumbai	300.0	Third Owner	24.0	135.0	Yamaha
8591	Yamaha YZF-R1 1000cc	1450000.0	Mumbai	340.0	First Owner	3.0	1000.0	Yamaha

	bike_name	price	city	kms_driven	owner	age	power	brand
5775	Yamaha FZ S V 2.0 150cc	65000.0	Pune	6914.0	First Owner	6.0	150.0	Yamaha

In [65]:

```
bullet = df[df['brand'] == "Royal Enfield"]
bullet
```

Out[65]:

	bike_name	price	city	kms_driven	owner	age	power	brand
1	Royal Enfield Classic 350cc	119900.0	Delhi	11000.0	First Owner	4.0	350.0	Royal Enfield
8	Royal Enfield Thunderbird X 350cc	145000.0	Bangalore	9190.0	First Owner	3.0	350.0	Royal Enfield
9	Royal Enfield Classic Desert Storm 500cc	88000.0	Delhi	19000.0	Second Owner	7.0	500.0	Royal Enfield
23	Royal Enfield Classic Chrome 500cc	121700.0	Kalyan	24520.0	First Owner	5.0	500.0	Royal Enfield
36	Royal Enfield Classic 350cc	98800.0	Kochi	39000.0	First Owner	5.0	350.0	Royal Enfield
...	...	...	...	...	...	...	...	...
9261	Royal Enfield Classic 500cc	146006.0	Guwahati	8575.0	First Owner	4.0	500.0	Royal Enfield
9319	Royal Enfield Classic 350cc	100000.0	Chennai	25000.0	First Owner	10.0	350.0	Royal Enfield
9337	Royal Enfield Himalayan 410cc	120000.0	Gurgaon	8492.0	First Owner	5.0	410.0	Royal Enfield
9338	Royal Enfield Himalayan 410cc	138000.0	Delhi	5000.0	First Owner	5.0	410.0	Royal Enfield
9344	Royal Enfield Bullet Twinspark 350cc	80000.0	Delhi	56968.0	First Owner	8.0	350.0	Royal Enfield

1346 rows × 8 columns

In [66]:

```
#The DataFrame bullet is filtered (bullet[bullet['city'] == "Jaipur"]) to select rows wh
ere the 'city' column equals "Jaipur".
#jaipur_bullet now contains only the rows where the bikes are located in Jaipur.

jaipur_bullet = bullet[bullet['city']== "Jaipur"]
```

In [67]:

```
#export csv file
jaipur_bullet.to_csv('jaipur_bullet.csv', index=False)
```

In [68]:

```
jaipur_bullet
```

Out[68]:

	bike_name	price	city	kms_driven	owner	age	power	brand
166	Royal Enfield Classic 350cc	123600.0	Jaipur	7702.0	First Owner	3.0	350.0	Royal Enfield
807	Royal Enfield Classic 350cc	115000.0	Jaipur	25000.0	First Owner	6.0	350.0	Royal Enfield
3009	Royal Enfield Himalayan 410cc	120000.0	Jaipur	14290.0	First Owner	5.0	410.0	Royal Enfield
3332	Royal Enfield Classic 350cc Dual Disc	120000.0	Jaipur	21000.0	First Owner	4.0	350.0	Royal Enfield
4497	Royal Enfield Classic 350cc	121000.0	Jaipur	20000.0	First Owner	4.0	350.0	Royal Enfield
4630	Royal Enfield Classic 350cc	90000.0	Jaipur	15437.0	First Owner	7.0	350.0	Royal Enfield
5193	Royal Enfield Electra 350cc	90000.0	Jaipur	14330.0	First Owner	3.0	350.0	Royal Enfield
6326	Royal Enfield Classic 350cc	120000.0	Jaipur	2069.0	First Owner	7.0	350.0	Royal Enfield
6655	Royal Enfield Classic Chrome 500cc	175000.0	Jaipur	7000.0	First Owner	5.0	500.0	Royal Enfield
6807	Royal Enfield Classic Chrome 500cc	160000.0	Jaipur	14000.0	First Owner	4.0	500.0	Royal Enfield
6897	Royal Enfield Thunderbird 350cc	135000.0	Jaipur	13500.0	First Owner	3.0	350.0	Royal Enfield

	bike_name	price	city	kms_driven	owner	age	power	brand
7137	Royal Enfield Classic 350cc	132000.0	Jaipur	3467.0	First Owner	6.0	350.0	Royal Enfield
7345	Royal Enfield Thunderbird 350cc	105000.0	Jaipur	1543.0	First Owner	5.0	350.0	Royal Enfield
7352	Royal Enfield Thunderbird 350cc	55000.0	Jaipur	4779.0	First Owner	11.0	350.0	Royal Enfield
7974	Royal Enfield Classic 350cc	114000.0	Jaipur	38000.0	First Owner	7.0	350.0	Royal Enfield
8067	Royal Enfield Thunderbird 350cc	128000.0	Jaipur	8500.0	First Owner	5.0	350.0	Royal Enfield
8100	Royal Enfield Classic 350cc	100000.0	Jaipur	22600.0	First Owner	8.0	350.0	Royal Enfield
8367	Royal Enfield Thunderbird 350cc	90000.0	Jaipur	9902.0	First Owner	7.0	350.0	Royal Enfield
8461	Royal Enfield Classic Squadron Blue 500cc	160000.0	Jaipur	8000.0	First Owner	5.0	500.0	Royal Enfield
8708	Royal Enfield Bullet Twinspark 350cc	90000.0	Jaipur	34548.0	First Owner	7.0	350.0	Royal Enfield
8818	Royal Enfield Thunderbird 500cc	120000.0	Jaipur	32814.0	Second Owner	8.0	500.0	Royal Enfield
9197	Royal Enfield Classic 350cc	90000.0	Jaipur	17000.0	First Owner	7.0	350.0	Royal Enfield
9198	Royal Enfield Bullet Electra 350cc	83800.0	Jaipur	21540.0	Second Owner	10.0	350.0	Royal Enfield

In [111]:

```
# Fetch these columns bike_name, price , age, kms_driven

jaipur_bullet[['bike_name','price','age']]
```

Out[111]:

	bike_name	price	age
166	Royal Enfield Classic 350cc	123600.0	3.0
807	Royal Enfield Classic 350cc	115000.0	6.0
3009	Royal Enfield Himalayan 410cc	120000.0	5.0
3332	Royal Enfield Classic 350cc Dual Disc	120000.0	4.0
4497	Royal Enfield Classic 350cc	121000.0	4.0
4630	Royal Enfield Classic 350cc	90000.0	7.0
5193	Royal Enfield Electra 350cc	90000.0	3.0
6326	Royal Enfield Classic 350cc	120000.0	7.0
6655	Royal Enfield Classic Chrome 500cc	175000.0	5.0
6807	Royal Enfield Classic Chrome 500cc	160000.0	4.0
6897	Royal Enfield Thunderbird 350cc	135000.0	3.0
7137	Royal Enfield Classic 350cc	132000.0	6.0
7345	Royal Enfield Thunderbird 350cc	105000.0	5.0
7352	Royal Enfield Thunderbird 350cc	55000.0	11.0
7974	Royal Enfield Classic 350cc	114000.0	7.0
8067	Royal Enfield Thunderbird 350cc	128000.0	5.0
8100	Royal Enfield Classic 350cc	100000.0	8.0
8367	Royal Enfield Thunderbird 350cc	90000.0	7.0
8461	Royal Enfield Classic Squadron Blue 500cc	160000.0	5.0
8708	Royal Enfield Bullet Twinspark 350cc	90000.0	7.0
8818	Royal Enfield Thunderbird 500cc	120000.0	8.0
9197	Royal Enfield Classic 350cc	90000.0	7.0
9198	Royal Enfield Bullet Electra 350cc	83800.0	10.0

In [70]:

```
#delete col. ==>owner
jaipur_bullet.drop('owner',axis='columns') # ,inplace = True
```

```
# or jaipur_bullet.drop('owner',axis='1')

#axis =>1 == columns
#axis = 0==> rows
```

Out[70]:

	bike_name	price	city	kms_driven	age	power	brand
166	Royal Enfield Classic 350cc	123600.0	Jaipur	7702.0	3.0	350.0	Royal Enfield
807	Royal Enfield Classic 350cc	115000.0	Jaipur	25000.0	6.0	350.0	Royal Enfield
3009	Royal Enfield Himalayan 410cc	120000.0	Jaipur	14290.0	5.0	410.0	Royal Enfield
3332	Royal Enfield Classic 350cc Dual Disc	120000.0	Jaipur	21000.0	4.0	350.0	Royal Enfield
4497	Royal Enfield Classic 350cc	121000.0	Jaipur	20000.0	4.0	350.0	Royal Enfield
4630	Royal Enfield Classic 350cc	90000.0	Jaipur	15437.0	7.0	350.0	Royal Enfield
5193	Royal Enfield Electra 350cc	90000.0	Jaipur	14330.0	3.0	350.0	Royal Enfield
6326	Royal Enfield Classic 350cc	120000.0	Jaipur	2069.0	7.0	350.0	Royal Enfield
6655	Royal Enfield Classic Chrome 500cc	175000.0	Jaipur	7000.0	5.0	500.0	Royal Enfield
6807	Royal Enfield Classic Chrome 500cc	160000.0	Jaipur	14000.0	4.0	500.0	Royal Enfield
6897	Royal Enfield Thunderbird 350cc	135000.0	Jaipur	13500.0	3.0	350.0	Royal Enfield
7137	Royal Enfield Classic 350cc	132000.0	Jaipur	3467.0	6.0	350.0	Royal Enfield
7345	Royal Enfield Thunderbird 350cc	105000.0	Jaipur	1543.0	5.0	350.0	Royal Enfield
7352	Royal Enfield Thunderbird 350cc	55000.0	Jaipur	4779.0	11.0	350.0	Royal Enfield
7974	Royal Enfield Classic 350cc	114000.0	Jaipur	38000.0	7.0	350.0	Royal Enfield
8067	Royal Enfield Thunderbird 350cc	128000.0	Jaipur	8500.0	5.0	350.0	Royal Enfield
8100	Royal Enfield Classic 350cc	100000.0	Jaipur	22600.0	8.0	350.0	Royal Enfield
8367	Royal Enfield Thunderbird 350cc	90000.0	Jaipur	9902.0	7.0	350.0	Royal Enfield
8461	Royal Enfield Classic Squadron Blue 500cc	160000.0	Jaipur	8000.0	5.0	500.0	Royal Enfield
8708	Royal Enfield Bullet Twinspark 350cc	90000.0	Jaipur	34548.0	7.0	350.0	Royal Enfield
8818	Royal Enfield Thunderbird 500cc	120000.0	Jaipur	32814.0	8.0	500.0	Royal Enfield
9197	Royal Enfield Classic 350cc	90000.0	Jaipur	17000.0	7.0	350.0	Royal Enfield
9198	Royal Enfield Bullet Electra 350cc	83800.0	Jaipur	21540.0	10.0	350.0	Royal Enfield

In [71]:

```
df
```

Out[71]:

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

7324 rows × 8 columns

In [114]:

```
#delete multiple col
#Coln--> owner, city, power

jaipur_bullet.drop(['owner','city','power']), axis='columns',inplace=True)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[114], line 4
      1 #delete multiple col
      2 #Coln--> owner, city, power
----> 4 jaipur_bullet.drop(('owner','city','power'), axis='columns',inplace=True)

File c:\Users\Krati\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\cor
e\frame.py:5581, in DataFrame.drop(self, labels, axis, index, columns, level, inplace, e
rrors)
   5433 def drop(
   5434     self,
   5435     labels: IndexLabel | None = None,
   (...)
   5442     errors: IgnoreRaise = "raise",
   5443 ) -> DataFrame | None:
   5444     """
   5445     Drop specified labels from rows or columns.
   5446
   (...)
   5579             weight  1.0     0.8
   5580     """
-> 5581     return super().drop(
   5582         labels=labels,
   5583         axis=axis,
   5584         index=index,
   5585         columns=columns,
   5586         level=level,
   5587         inplace=inplace,
   5588         errors=errors,
   5589     )

File c:\Users\Krati\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\cor
e\generic.py:4788, in NDFrame.drop(self, labels, axis, index, columns, level, inplace, e
rrors)
   4786 for axis, labels in axes.items():
   4787     if labels is not None:
-> 4788         obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   4790 if inplace:
   4791     self._update_inplace(obj)

File c:\Users\Krati\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\cor
e\generic.py:4830, in NDFrame._drop_axis(self, labels, axis, level, errors, only_slice)
   4828         new_axis = axis.drop(labels, level=level, errors=errors)
   4829     else:
-> 4830         new_axis = axis.drop(labels, errors=errors)
   4831     indexer = axis.get_indexer(new_axis)
   4833 # Case for non-unique axis
   4834 else:

File c:\Users\Krati\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\cor
e\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: "[('owner', 'city', 'power')] not found in axis"
```

In [73]:

```
jaipur_bullet
```

Out[73]:

	bike_name	price	kms_driven	age	brand
166	Royal Enfield Classic 350cc	123600.0	7702.0	3.0	Royal Enfield
807	Royal Enfield Classic 350cc	115000.0	25000.0	6.0	Royal Enfield
3009	Royal Enfield Himalayan 410cc	120000.0	14290.0	5.0	Royal Enfield
3332	Royal Enfield Classic 350cc Dual Disc	120000.0	21000.0	4.0	Royal Enfield
4497	Royal Enfield Classic 350cc	121000.0	20000.0	4.0	Royal Enfield
4630	Royal Enfield Classic 350cc	90000.0	15437.0	7.0	Royal Enfield
5193	Royal Enfield Electra 350cc	90000.0	14330.0	3.0	Royal Enfield
6326	Royal Enfield Classic 350cc	120000.0	2069.0	7.0	Royal Enfield
6655	Royal Enfield Classic Chrome 500cc	175000.0	7000.0	5.0	Royal Enfield
6807	Royal Enfield Classic Chrome 500cc	160000.0	14000.0	4.0	Royal Enfield
6897	Royal Enfield Thunderbird 350cc	135000.0	13500.0	3.0	Royal Enfield
7137	Royal Enfield Classic 350cc	132000.0	3467.0	6.0	Royal Enfield
7345	Royal Enfield Thunderbird 350cc	105000.0	1543.0	5.0	Royal Enfield
7352	Royal Enfield Thunderbird 350cc	55000.0	4779.0	11.0	Royal Enfield
7974	Royal Enfield Classic 350cc	114000.0	38000.0	7.0	Royal Enfield
8067	Royal Enfield Thunderbird 350cc	128000.0	8500.0	5.0	Royal Enfield
8100	Royal Enfield Classic 350cc	100000.0	22600.0	8.0	Royal Enfield
8367	Royal Enfield Thunderbird 350cc	90000.0	9902.0	7.0	Royal Enfield
8461	Royal Enfield Classic Squadron Blue 500cc	160000.0	8000.0	5.0	Royal Enfield
8708	Royal Enfield Bullet Twinspark 350cc	90000.0	34548.0	7.0	Royal Enfield
8818	Royal Enfield Thunderbird 500cc	120000.0	32814.0	8.0	Royal Enfield
9197	Royal Enfield Classic 350cc	90000.0	17000.0	7.0	Royal Enfield
9198	Royal Enfield Bullet Electra 350cc	83800.0	21540.0	10.0	Royal Enfield

filtering------- excel:- .csv read data | | ------>>read in pandas

In [74]:

```
import pandas as pd
import numpy as np
```

In [75]:

```
df = pd.read_csv('Used_Bikes.csv')
df
```

Out[75]:

	bike_name	price	city	kms_driven	owner	age	power	brand
0	TVS Star City Plus Dual Tone 110cc	35000.0	Ahmedabad	17654.0	First Owner	3.0	110.0	TVS
1	Royal Enfield Classic 350cc	119900.0	Delhi	11000.0	First Owner	4.0	350.0	Royal Enfield
2	Triumph Daytona 675R	600000.0	Delhi	110.0	First Owner	8.0	675.0	Triumph
3	TVS Apache RTR 180cc	65000.0	Bangalore	16329.0	First Owner	4.0	180.0	TVS
4	Yamaha FZ S V 2.0 150cc-Ltd. Edition	80000.0	Bangalore	10000.0	First Owner	3.0	150.0	Yamaha
...	...	...	...	...	...	...	...	...
32643	Hero Passion Pro 100cc	39000.0	Delhi	22000.0	First Owner	4.0	100.0	Hero

	bike_name	price	city	kms_driven	owner	age	power	brand
32644	TVS Apache RTR 180cc	30000.0	Karnal	6639.0	First Owner	9.0	180.0	TVS
32645	Bajaj Avenger Street 220	60000.0	Delhi	20373.0	First Owner	6.0	220.0	Bajaj
32646	Hero Super Splendor 125cc	15600.0	Jaipur	84186.0	First Owner	16.0	125.0	Hero
32647	Bajaj Pulsar 150cc	22000.0	Pune	60857.0	First Owner	13.0	150.0	Bajaj

32648 rows × 8 columns

In [76]:

```
data = {'A':[2,5,np.nan,8,np.nan,9],
        'B':[np.nan,45,np.nan,89,63,np.nan],
        'C':[np.nan,74,np.nan,np.nan,85,4],
        'D':[10,20,30,40,50,60]}
data
```

Out[76]:

```
{'A': [2, 5, nan, 8, nan, 9],
 'B': [nan, 45, nan, 89, 63, nan],
 'C': [nan, 74, nan, nan, 85, 4],
 'D': [10, 20, 30, 40, 50, 60]}
```

NaN-> none/ missing value

In [77]:

```
df2=pd.DataFrame(data)
df2
```

Out[77]:

	A	B	C	D
0	2.0	NaN	NaN	10
1	5.0	45.0	74.0	20
2	NaN	NaN	NaN	30
3	8.0	89.0	NaN	40
4	NaN	63.0	85.0	50
5	9.0	NaN	4.0	60

ML-> missing value not accept

Missing value Handling 1- Remove the records ---> if data 3-4% out of total is missing then we use 2- fill the records --->

In [78]:

```
df2.dropna()  # by default remove all those rows that contains missing values
#df2.dropna(inplace=True)----> changes done permanently
```

Out[78]:

	A	B	C	D
1	5.0	45.0	74.0	20

In [79]:

```
df2.dropna(axis=1) #(axis='columns)
```

Out[79]:

	D

	D
0	10
1	20
2	30
3	40
4	50
5	60

In [80]:

```
# Check for missing values
df2.isnull().sum()
```

Out[80]:

```
A    2
B    3
C    3
D    0
dtype: int64
```

In [81]:

```
df2
```

Out[81]:

	A	B	C	D
0	2.0	NaN	NaN	10
1	5.0	45.0	74.0	20
2	NaN	NaN	NaN	30
3	8.0	89.0	NaN	40
4	NaN	63.0	85.0	50
5	9.0	NaN	4.0	60

In [82]:

```
# Compute and print the total number of missing values
print("Total missing values in your df: ",df2.isnull().sum().sum())
```

```
Total missing values in your df:  8
```

In [83]:

```
# Get the number of rows in the DataFrame
df2.shape[0]  # total rows
```

Out[83]:

```
6
```

In [84]:

```
df2.isnull().sum()/df2.shape[0]*100 #--> find in percentage
```

Out[84]:

```
A    33.333333
B    50.000000
C    50.000000
D     0.000000
dtype: float64
```

Filling the records

check column ...->1.numerical -----> constant -----> mean avg -----> mr=edian ->2.categorical -----> constant ----> mode

In [85]:

```
df2.fillna(500) #missing value replace by 500
```

Out[85]:

	A	B	C	D
0	2.0	500.0	500.0	10
1	5.0	45.0	74.0	20
2	500.0	500.0	500.0	30
3	8.0	89.0	500.0	40
4	500.0	63.0	85.0	50
5	9.0	500.0	4.0	60

In [86]:

```
df2
```

Out[86]:

	A	B	C	D
0	2.0	NaN	NaN	10
1	5.0	45.0	74.0	20
2	NaN	NaN	NaN	30
3	8.0	89.0	NaN	40
4	NaN	63.0	85.0	50
5	9.0	NaN	4.0	60

In [87]:

```
# A=500
# B=600
# C=700

df2['A'].fillna(500)
```

Out[87]:

```
0      2.0
1      5.0
2    500.0
3      8.0
4    500.0
5      9.0
Name: A, dtype: float64
```

In [88]:

```
df2['B'].fillna(600)
```

Out[88]:

```
0    600.0
1     45.0
2    600.0
3     89.0
4     63.0
5    600.0
Name: B, dtype: float64
```

In [89]:

```
df2['C'].fillna(700)
```

Out[89]:

```
0    700.0
1     74.0
2    700.0
3    700.0
4     85.0
5      4.0
Name: C, dtype: float64
```

In [90]:

```
df2['A'].mean()
```

Out[90]:

```
np.float64(6.0)
```

In [91]:

```
df2['A'].fillna(6.0)
#or
#df2['A'].fillna(df2['A'].mean())
```

Out[91]:

```
0    2.0
1    5.0
2    6.0
3    8.0
4    6.0
5    9.0
Name: A, dtype: float64
```

In [92]:

```
df2['A'].fillna(df2['A'].median())
```

Out[92]:

```
0    2.0
1    5.0
2    6.5
3    8.0
4    6.5
5    9.0
Name: A, dtype: float64
```

In [93]:

```
#df2['A'].mode() ------>not sure
```

Groupby

In [94]:

```
df
```

Out[94]:

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

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

32648 rows × 8 columns

In [95]:

```
# Find the minimum price
df['price'].min()
```

Out[95]:

```
np.float64(4400.0)
```

In [96]:

```
# Find the maximum price
df['price'].max()
```

Out[96]:

```
np.float64(1900000.0)
```

In [97]:

```
# Find the mean price
df['price'].mean()
```

Out[97]:

```
np.float64(68295.41763660868)
```

In [98]:

```
df[df['brand']=="TVS"]['price'].min()
```

Out[98]:

```
np.float64(5800.0)
```

In [99]:

```
df['brand'].unique()
```

Out[99]:

```
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)
```

In [100]:

```
# Group by 'brand'
brand_group = df.groupby('brand')
brand_group
```

Out[100]:

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001EF5F3B0AD0>
```

In [101]:

```
brand_group.get_group('TVS')
```

Out[101]:

	bike_name	price	city	kms_driven	owner	age	power	brand
0	TVS Star City Plus Dual Tone 110cc	35000.0	Ahmedabad	17654.0	First Owner	3.0	110.0	TVS
3	TVS Apache RTR 180cc	65000.0	Bangalore	16329.0	First Owner	4.0	180.0	TVS
52	TVS Apache RTR 160cc	60000.0	Mumbai	30000.0	First Owner	5.0	160.0	TVS
114	TVS Apache RTR 160 4V Disc	69900.0	Delhi	8700.0	First Owner	3.0	160.0	TVS
130	TVS Phoenix Disc 125cc	21500.0	Barasat	10500.0	First Owner	5.0	125.0	TVS
...	...	...	...	...	...	...	...	...
32549	TVS Apache RTR 180cc	30000.0	Karnal	6639.0	First Owner	9.0	180.0	TVS
32568	TVS Apache RTR 180cc	30000.0	Karnal	6639.0	First Owner	9.0	180.0	TVS
32587	TVS Apache RTR 180cc	30000.0	Karnal	6639.0	First Owner	9.0	180.0	TVS
32606	TVS Apache RTR 180cc	30000.0	Karnal	6639.0	First Owner	9.0	180.0	TVS
32644	TVS Apache RTR 180cc	30000.0	Karnal	6639.0	First Owner	9.0	180.0	TVS

1247 rows × 8 columns

In [102]:

```
# Retrieve the group where the brand is 'Triumph'
brand_group.get_group('Triumph')
```

Out[102]:

	bike_name	price	city	kms_driven	owner	age	power	brand
2	Triumph Daytona 675R	600000.0	Delhi	110.0	First Owner	8.0	675.0	Triumph
93	Triumph Street Triple 765	790000.0	Mumbai	19000.0	First Owner	3.0	765.0	Triumph
284	Triumph Street Triple ABS 675cc	599999.0	Mumbai	7800.0	Third Owner	5.0	675.0	Triumph
614	Triumph Speed Triple 1050cc	750000.0	Koppal	9500.0	First Owner	4.0	1050.0	Triumph
1319	Triumph Tiger 800 XRX	750000.0	Delhi	10000.0	First Owner	6.0	800.0	Triumph
1362	Triumph Thunderbird Storm 1700cc	700000.0	Guwahati	176.0	First Owner	5.0	1700.0	Triumph
1375	Triumph Tiger 800 XCA	1300000.0	Ludhiana	9500.0	First Owner	4.0	800.0	Triumph
1440	Triumph Tiger 800 XCA	1300000.0	Ludhiana	9500.0	First Owner	4.0	800.0	Triumph
1867	Triumph Street Twin 900cc	790000.0	Delhi	2500.0	First Owner	3.0	900.0	Triumph
1927	Triumph Street Triple 765	860000.0	Udaipur	2100.0	First Owner	3.0	765.0	Triumph
1998	Triumph Street Triple 765	860000.0	Udaipur	2100.0	First Owner	3.0	765.0	Triumph
2107	Triumph Street Twin 900cc	790000.0	Delhi	2500.0	First Owner	3.0	900.0	Triumph
2157	Triumph Tiger 800 XR	1170000.0	Pune	31500.0	First Owner	5.0	800.0	Triumph
2244	Triumph Bonneville T100 865cc	550000.0	Bangalore	25536.0	First Owner	7.0	865.0	Triumph
3007	Triumph Thruxton R 1200cc	1000000.0	Delhi	1500.0	First Owner	4.0	1200.0	Triumph
3054	Triumph Thruxton R 1200cc	1000000.0	Delhi	1500.0	First Owner	4.0	1200.0	Triumph
3095	Triumph Street Triple ABS 675cc	650000.0	Mumbai	16000.0	First Owner	4.0	675.0	Triumph
3138	Triumph Bonneville T100 900cc	640000.0	Delhi	7500.0	Second Owner	4.0	900.0	Triumph
4597	Triumph Bonneville T100 865cc	500000.0	Delhi	5600.0	Second Owner	6.0	865.0	Triumph
4662	Triumph Bonneville T100 865cc	500000.0	Delhi	5600.0	Second Owner	6.0	865.0	Triumph
4828	Triumph Thruxton R 1200cc	950000.0	Delhi	4600.0	First Owner	4.0	1200.0	Triumph
4917	Triumph Speed Triple 1050cc	900000.0	Bangalore	8000.0	Second Owner	5.0	1050.0	Triumph
6793	Triumph Street Triple 765	789000.0	Pune	13000.0	First Owner	4.0	765.0	Triumph
6794	Triumph Tiger 800 XCA	1200000.0	Hyderabad	25800.0	First Owner	4.0	800.0	Triumph
6795	Triumph Street Triple 765	875000.0	Rajkot	750.0	First Owner	2.0	765.0	Triumph

	bike_name	price	city	kms_driven	owner	age	power	brand
7621	Triumph Daytona 675 ABS	699000.0	Delhi	9000.0	Third Owner	5.0	675.0	Triumph

In [103]:

```
brand_group[['price']].min()
```

Out[103]:

brand	price
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

In [104]:

```
brand_group[['price']].max()
```

Out[104]:

brand	price
BMW	1800000.0
Bajaj	195000.0
Benelli	785000.0
Ducati	1500000.0
Harley-Davidson	1100000.0
Hero	104000.0
Honda	800000.0
Hyosung	403500.0

	price
Hyosung	493500.0
Ideal	100000.0
brand	
Indian	1900000.0
Jawa	223000.0
KTM	860000.0
Kawasaki	1100000.0
LML	4400.0
MV	1500000.0
Mahindra	175000.0
Rajdoot	75000.0
Royal Enfield	285000.0
Suzuki	1260000.0
TVS	224000.0
Triumph	1300000.0
Yamaha	1550000.0
Yezdi	68000.0

In [105]:

```
# Apply multiple aggregations to the 'price' column
brand_group['price'].agg(min_price='min',max_price='max',avg_price='mean')
```

Out[105]:

brand	min_price	max_price	avg_price
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** **min_price**	**68000.0** **max_price**	**6.800000e+04** **avg_price**

In [106]:

```
brand_group['kms_driven'].agg(min_dis='min',max_dis='max')
```

Out[106]:

brand	min_dis	max_dis
BMW	2200.0	18000.0
Bajaj	22.0	717794.0
Benelli	1000.0	27000.0
Ducati	3.0	31000.0
Harley-Davidson	200.0	63000.0
Hero	1.0	750000.0
Honda	68.0	654984.0
Hyosung	210.0	51791.0
Ideal	80000.0	80000.0
Indian	172.0	1700.0
Jawa	381.0	2388.0
KTM	40.0	100000.0
Kawasaki	180.0	36102.0
LML	55000.0	55000.0
MV	1911.0	6898.0
Mahindra	1400.0	55000.0
Rajdoot	500.0	500.0
Royal Enfield	40.0	300965.0
Suzuki	183.0	85000.0
TVS	100.0	266028.0
Triumph	110.0	31500.0
Yamaha	45.0	566931.0
Yezdi	23.0	23.0

In [107]:

```
df['owner'].value_counts()
```

Out[107]:

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

In [108]:

```
brand_group = df.groupby('owner')
brand_group
```

Out[108]:

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001EF601922A0>
```

In [109]:

```
brand_group[['kms_driven'].agg(min_dis='min',max_dis='max')][['price'].agg(min_price='min',max_price='max',avg_price='mean')]
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[109], line 1
----> 1 brand_group[['kms_driven'].agg(min_dis='min',max_dis='max')][['price'].agg(min_price='min',max_price='max',avg_price='mean')]

AttributeError: 'list' object has no attribute 'agg'
```

In [ ]:

```
df.select_dtypes(include='O')
```

Out[ ]:

	bike_name	city	owner	brand
0	TVS Star City Plus Dual Tone 110cc	Ahmedabad	First Owner	TVS
1	Royal Enfield Classic 350cc	Delhi	First Owner	Royal Enfield
2	Triumph Daytona 675R	Delhi	First Owner	Triumph
3	TVS Apache RTR 180cc	Bangalore	First Owner	TVS
4	Yamaha FZ S V 2.0 150cc-Ltd. Edition	Bangalore	First Owner	Yamaha
...	...	...	...	...
32643	Hero Passion Pro 100cc	Delhi	First Owner	Hero
32644	TVS Apache RTR 180cc	Karnal	First Owner	TVS
32645	Bajaj Avenger Street 220	Delhi	First Owner	Bajaj
32646	Hero Super Splendor 125cc	Jaipur	First Owner	Hero
32647	Bajaj Pulsar 150cc	Pune	First Owner	Bajaj

32648 rows × 4 columns

In [ ]:

```
cat_col=df.select_dtypes(include='O')
cat_col
```

Out[ ]:

	bike_name	city	owner	brand
0	TVS Star City Plus Dual Tone 110cc	Ahmedabad	First Owner	TVS
1	Royal Enfield Classic 350cc	Delhi	First Owner	Royal Enfield
2	Triumph Daytona 675R	Delhi	First Owner	Triumph
3	TVS Apache RTR 180cc	Bangalore	First Owner	TVS
4	Yamaha FZ S V 2.0 150cc-Ltd. Edition	Bangalore	First Owner	Yamaha
...	...	...	...	...
32643	Hero Passion Pro 100cc	Delhi	First Owner	Hero
32644	TVS Apache RTR 180cc	Karnal	First Owner	TVS
32645	Bajaj Avenger Street 220	Delhi	First Owner	Bajaj
32646	Hero Super Splendor 125cc	Jaipur	First Owner	Hero
32647	Bajaj Pulsar 150cc	Pune	First Owner	Bajaj

32648 rows × 4 columns

In [ ]:

```
df
```

Out[ ]:

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

32648 rows × 8 columns

In [ ]:

```
num_col = df.select_dtypes(exclude='O')
num_col.head()
```

Out[ ]:

	price	kms_driven	age	power
0	35000.0	17654.0	3.0	110.0
1	119900.0	11000.0	4.0	350.0
2	600000.0	110.0	8.0	675.0
3	65000.0	16329.0	4.0	180.0
4	80000.0	10000.0	3.0	150.0

In [ ]:

```
pd.concat([cat_col,num_col],axis='columns')
```

Out[ ]:

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

32648 rows × 8 columns

In [ ]:

```
pd.merge
```

Out[ ]:

```
<function pandas.core.reshape.merge.merge(left: 'DataFrame | Series', right: 'DataFrame
| Series', how: 'MergeHow' = 'inner', on: 'IndexLabel | AnyArrayLike | None' = None, lef
t_on: 'IndexLabel | AnyArrayLike | None' = None, right_on: 'IndexLabel | AnyArrayLike |
None' = None, left_index: 'bool' = False, right_index: 'bool' = False, sort: 'bool' = Fa
lse, suffixes: 'Suffixes' = ('_x', '_y'), copy: 'bool | None' = None, indicator: 'str |
bool' = False, validate: 'str | None' = None) -> 'DataFrame'>
```

In [ ]:

```

```