NumPy

1. NumPy Basic Ops

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

-> Create a 1D Array from 1 to 20.

In [186…
```
array_1d = np.arange(1, 21)
print("1D Array from 1 to 20:\n", array_1d)
```

```
1D Array from 1 to 20:
 [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20]
```

-> Create a 3×4 matrix of ones and reshape it to 4×3

In [189…
```
mat_1 = np.ones((3,4))
print("\n 3x4 Matrix of Ones: \n", mat_1)

# Reshapping It to 4x3

reshape = mat_1.reshape((4,3))
print("\n 4x3 Reshaped Materix of Ones: \n", reshape)
```

```
3x4 Matrix of Ones:
[[1. 1. 1. 1.]
[1. 1. 1. 1.]
[1. 1. 1. 1.]]

4x3 Reshaped Materix of Ones:
[[1. 1. 1.]
[1. 1. 1.]
[1. 1. 1.]
[1. 1. 1.]]
```

-> Create a 5×5 identity matrix

In [194…
```
i5 = np.eye(5)
print("\n 5x5 Identity Matrix: \n", i5)
```

```
5x5 Identity Matrix:
[[1. 0. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 0. 1. 0. 0.]
[0. 0. 0. 1. 0.]
[0. 0. 0. 0. 1.]]
```

-> Generate 15 equally spaced numbers between 5 and 50.

In [199…
```
e_nums = np.linspace(5,50,15)
print("\n 15 equally spaced nums between 5 and 15: \n", e_nums)
```

```
15 equally spaced nums between 5 and 15:
[ 5.          8.21428571 11.42857143 14.64285714 17.85714286 21.07142857
24.28571429 27.5        30.71428571 33.92857143 37.14285714 40.35714286
43.57142857 46.78571429 50.        ]
```

-> Generate a 4×4 matrix of random integers between 1 and 100.

In [201…
```
r_nums = np.random.randint(1,101,size=(4,4))
print("\n 4x4 matrix of Random int between 1 & 100 \n",r_nums)
```

```
4x4 matrix of Random int between 1 & 100
[[48 17 75 76]
[47 94 95 48]
[36 68 89 44]
[93 58 61 56]]
```

2. NumPy Array Ops

-> Create a 3×3 matrix of random integers between 1 and 100

In [205…
```
r_mat = np.random.randint(1,100,size=(3,3))
print("\n 3x3 Matrix of random nums \n",r_mat)
```

```
3x3 Matrix of random nums
[[34 34 99]
[14 14 29]
[37 85  1]]
```

-> Extract: First row, Second column, Center element.

In [208…
```
fr = r_mat[0]
print("\n First Row: \n",fr)

sc = r_mat[:,1]
print("\n Second Column \n",sc)

se = r_mat[1,1]
print("\n Center Element \n",se)
```

```
First Row:
[34 34 99]

Second Column
[34 14 85]

Center Element
14
```

-> Replace all values greater than 50 in a matrix with 999.

In [211…
```
r_mat_c = r_mat.copy()
r_mat_c[r_mat_c > 50] = 999
print("\n Replaced Matrix \n",r_mat_c)
```

```
Replaced Matrix
[[ 34  34 999]
[ 14  14  29]
[ 37 999   1]]
```

-> Multiply a 1D array of size 5 (random integers between 1 to 10) by 10 using broadcasting.

In [214…
```
arr_1d = np.random.randint(1,11,size=5)
multi_arr = arr_1d * 10
print("\n Original 1D Array: \n",arr_1d)
print("\n Multiplied Array: \n",multi_arr)
```

```
Original 1D Array:
[10  9  3  8  6]

Multiplied Array:
[100  90  30  80  60]
```

3. Math & Stats Ops

->Create a 3×3 matrix of random integers between 1 and 100

In [225…
```
r_mat = np.random.randint(1,100,size=(3,3))
print("\n 3x3 Matrix of random nums \n",r_mat)
```

```
3x3 Matrix of random nums
[[24  1 95]
[79 17 61]
[99 93 46]]
```

-> Compute sum, mean, median, std, var, min, and max of the above array.

In [230…
```
print("\nStatistics:")
print("Sum:", np.sum(r_mat))
print("Mean:", np.mean(r_mat))
```

```
print("Median:", np.median(r_mat))
print("Standard Deviation:", np.std(r_mat))
print("Variance:", np.var(r_mat))
print("Minimum:", np.min(r_mat))
print("Maximum:", np.max(r_mat))
```

```
Statistics:
Sum: 515
Mean: 57.22222222222222
Median: 61.0
Standard Deviation: 34.880041341010724
Variance: 1216.6172839506173
Minimum: 1
Maximum: 99
```

-> Normalize a 1D array of size 5 (random integers between 1 to 10) to scale values between 0 and 1.

In [234…

```
arr_1d = np.random.randint(1,11,size=5)
print("\n 1D Array: \n",arr_1d)
min = np.min(arr_1d)
max = np.max(arr_1d)
norm_arr = ((arr_1d - min) / (max - min) )
print("\n Normalized Array: \n",norm_arr)
```

```
1D Array:
[ 8 10  5  1  4]

Normalized Array:
[0.77777778 1.         0.44444444 0.         0.33333333]
```

4. NumPy Matrix Ops & Algebra

-> Generate Following two NumPy matrix import numpy as np

A = ([4, 2], [1, 3])

B = ([2, 0], [1, 5])

In [240…

```
A = np.array( [[4, 2], [1, 3]] )
B = np.array( [[2, 0], [1, 5]] )

print("\n A: \n",A)
print("\n B: \n",B)
```

```
A:
[[4 2]
[1 3]]

B:
[[2 0]
[1 5]]
```

-> Matrix Multiplication

In [252…
```
mat_multi = np.matmul(A,B)
print("\n Multiplication of A & B: \n",mat_multi)
```

```
Multiplication of A & B:
[[10 10]
[ 5 15]]
```

-> dot product of A and B.

In [254…
```
dot = np.dot(A,B)
print("\n dot product of A & B.: \n",dot)
```

```
dot product of A & B.:
[[10 10]
[ 5 15]]
```

-> Element wise addition/ subtraction/ multiplications/ division of A and B

In [250…
```
add = A + B
sub = A - B
multi = A * B
div = A / B

print("\nElement-wise Addition:\n", add)
print("Element-wise Subtraction:\n", sub)
print("Element-wise Multiplication:\n", multi)
print("Element-wise Division:\n", div)
```

```
Element-wise Addition:
 [[6 2]
 [2 8]]
Element-wise Subtraction:
 [[ 2  2]
 [ 0 -2]]
Element-wise Multiplication:
 [[ 8  0]
 [ 1 15]]
Element-wise Division:
 [[2.  inf]
 [1.  0.6]]
C:\Users\rajva\AppData\Local\Temp\ipykernel_18728\3905969198.py:4: RuntimeWarning: d
ivide by zero encountered in divide
  div = A / B
```

-> Transpose of A

In [277…
```
trans_A = A.T
print("\n Transpose of A: \n",trans_A)
```

```
Transpose of A:
[[4 1]
[2 3]]
```

-> Determinant of A

In [271…
```
det_A = np.linalg.det(A)
print("\n Determinant of A: \n",det_A)
```

```
Determinant of A:
10.000000000000002
```

-> Inverse of A

In [269…
```
if det_A != 0:
    inv_A = np.linalg.inv(A)
    print("\nInverse of A:\n", inv_A)
else:
    print("\nMatrix A cannot be inverted.")
```

```
Inverse of A:
 [[ 0.3 -0.2]
 [-0.1  0.4]]
```

-> Eigenvalues and Eigenvectors of A

In [264…
```
eigenvalues= np.linalg.eig(A)
eigenvectors = np.linalg.eig(A)
print("\nEigenvalues of A:\n", eigenvalues)
print("\nEigenvectors of A:\n", eigenvectors)
```

```
Eigenvalues of A:
 EigResult(eigenvalues=array([5., 2.]), eigenvectors=array([[ 0.89442719, -0.7071067
8],
       [ 0.4472136 ,  0.70710678]]))

Eigenvectors of A:
 EigResult(eigenvalues=array([5., 2.]), eigenvectors=array([[ 0.89442719, -0.7071067
8],
       [ 0.4472136 ,  0.70710678]]))
```

-> Solve System of Equations:

2x + y = 8

3x + 4y = 18

In [261…
```
coeff = np.array([[2,1],[3,4]])
cons = np.array([8,18])

sol = np.linalg.solve(coeff, cons)
print("\n Solution of the Equations x & y : ",sol)
```

```
Solution of the Equations x & y :  [2.8 2.4]
```

Pandas Exercise

1. Series & DataFrame Basics

-> Given the following list of marks:

[78, 85, 92, 70, 66]

Create a Pandas Series and assign the following student names as indices:

['Amit', 'Bhavna', 'Chetan', 'Divya', 'Esha']

Display the Series

In [ ]:
```
marks = [78, 85, 92, 70, 66]
names = ['Amit', 'Bhavna', 'Chetan', 'Divya', 'Esha']

marks_series = pd.Series(data=marks,index=names)
print("\nPandas Series : \n",marks_series)
```

-> Create a Pandas DataFrame and display:

The full DataFrame

The column names

The shape of the DataFrame

In [285…
```
# Create DataFrame
data = {
    'Name': ['Amit', 'Bhavna', 'Chetan', 'Divya', 'Esha'],
    'Gender': ['Male', 'Female', 'Male', 'Female', 'Female'],
    'Math': [78, 85, 92, 70, 66],
    'Science': [88, 79, 95, 72, 60]
}

df = pd.DataFrame(data)

print("\nFull DataFrame:\n")
print(df)
```

```
# Display column names
print("\nColumn Names:\n")
print(df.columns.tolist())

# Display shape (rows, columns)
print("\nShape of the DataFrame:\n")
print(df.shape)
```

```
Full DataFrame:

     Name  Gender  Math  Science
0    Amit    Male    78       88
1  Bhavna  Female    85       79
2  Chetan    Male    92       95
3   Divya  Female    70       72
4    Esha  Female    66       60

Column Names:

['Name', 'Gender', 'Math', 'Science']

Shape of the DataFrame:

(5, 4)
```

2.Data Exploration

-> Load the dataset from this url "https://archive.ics.uci.edu/ml/machine-learningdatabases/autos/imports-85.data"

In [7]:
```
import pandas as pd
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.d
df = pd.read_csv(url, header=None)
print(df.head())
```

```
  0    1            2    3    4     5            6    7      8     9   ...  \
0   3    ?  alfa-romero  gas  std   two  convertible  rwd  front  88.6  ...
1   3    ?  alfa-romero  gas  std   two  convertible  rwd  front  88.6  ...
2   1    ?  alfa-romero  gas  std   two    hatchback  rwd  front  94.5  ...
3   2  164         audi  gas  std  four        sedan  fwd  front  99.8  ...
4   2  164         audi  gas  std  four        sedan  4wd  front  99.4  ...

    16    17    18    19    20   21    22  23  24     25
0  130  mpfi  3.47  2.68   9.0  111  5000  21  27  13495
1  130  mpfi  3.47  2.68   9.0  111  5000  21  27  16500
2  152  mpfi  2.68  3.47   9.0  154  5000  19  26  16500
3  109  mpfi  3.19  3.40  10.0  102  5500  24  30  13950
4  136  mpfi  3.19  3.40   8.0  115  5500  18  22  17450

[5 rows x 26 columns]
```

-> Assign names of columns:

["symboling","normalized-losses","make","fuel-type","aspiration", "num-ofdoors","body-style", "drive-wheels","engine-ocation","wheelbase","length","width","height","curb-weight","engine-type","num-of-cylinders","engine-size","fuel-system","bore","stroke","compression-ratio","horsepower","peak-rpm","city-mpg","highway-mpg","price"]

In [296…
```
df.columns = ["symboling","normalized-losses","make","fuel-type","aspiration", "num
print(df.head())
```

```
  symboling normalized-losses         make fuel-type aspiration num-of-doors  \
0         3                 ?  alfa-romero       gas        std          two
1         3                 ?  alfa-romero       gas        std          two
2         1                 ?  alfa-romero       gas        std          two
3         2               164         audi       gas        std         four
4         2               164         audi       gas        std         four

    body-style drive-wheels engine-location  wheel-base  ...  engine-size  \
0  convertible          rwd           front        88.6  ...          130
1  convertible          rwd           front        88.6  ...          130
2    hatchback          rwd           front        94.5  ...          152
3        sedan          fwd           front        99.8  ...          109
4        sedan          4wd           front        99.4  ...          136

   fuel-system  bore  stroke compression-ratio horsepower  peak-rpm city-mpg  \
0         mpfi  3.47    2.68               9.0        111      5000       21
1         mpfi  3.47    2.68               9.0        111      5000       21
2         mpfi  2.68    3.47               9.0        154      5000       19
3         mpfi  3.19    3.40              10.0        102      5500       24
4         mpfi  3.19    3.40               8.0        115      5500       18

  highway-mpg  price
0          27  13495
1          27  16500
2          26  16500
3          30  13950
4          22  17450

[5 rows x 26 columns]
```

-> Display .shape, .columns, .info(), and .describe()

In [300…
```
print(df.shape)
print(df.columns)
print(df.info())
print(df.describe())
```

```
(205, 26)
Index(['symboling', 'normalized-losses', 'make', 'fuel-type', 'aspiration',
       'num-of-doors', 'body-style', 'drive-wheels', 'engine-location',
       'wheel-base', 'length', 'width', 'height', 'curb-weight', 'engine-type',
       'num-of-cylinders', 'engine-size', 'fuel-system', 'bore', 'stroke',
       'compression-ratio', 'horsepower', 'peak-rpm', 'city-mpg',
       'highway-mpg', 'price'],
      dtype='object')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   symboling          205 non-null    int64
 1   normalized-losses  205 non-null    object
 2   make               205 non-null    object
 3   fuel-type          205 non-null    object
 4   aspiration         205 non-null    object
 5   num-of-doors       205 non-null    object
 6   body-style         205 non-null    object
 7   drive-wheels       205 non-null    object
 8   engine-location    205 non-null    object
 9   wheel-base         205 non-null    float64
 10  length             205 non-null    float64
 11  width              205 non-null    float64
 12  height             205 non-null    float64
 13  curb-weight        205 non-null    int64
 14  engine-type        205 non-null    object
 15  num-of-cylinders   205 non-null    object
 16  engine-size        205 non-null    int64
 17  fuel-system        205 non-null    object
 18  bore               205 non-null    object
 19  stroke             205 non-null    object
 20  compression-ratio  205 non-null    float64
 21  horsepower         205 non-null    object
 22  peak-rpm           205 non-null    object
 23  city-mpg           205 non-null    int64
 24  highway-mpg        205 non-null    int64
 25  price              205 non-null    object
dtypes: float64(5), int64(5), object(16)
memory usage: 41.8+ KB
None
        symboling  wheel-base      length       width      height  \
count  205.000000  205.000000  205.000000  205.000000  205.000000
mean     0.834146   98.756585  174.049268   65.907805   53.724878
std      1.245307    6.021776   12.337289    2.145204    2.443522
min     -2.000000   86.600000  141.100000   60.300000   47.800000
25%      0.000000   94.500000  166.300000   64.100000   52.000000
50%      1.000000   97.000000  173.200000   65.500000   54.100000
75%      2.000000  102.400000  183.100000   66.900000   55.500000
max      3.000000  120.900000  208.100000   72.300000   59.800000

       curb-weight  engine-size  compression-ratio    city-mpg  highway-mpg
count   205.000000   205.000000         205.000000  205.000000   205.000000
mean   2555.565854   126.907317          10.142537   25.219512    30.751220
std     520.680204    41.642693           3.972040    6.542142     6.886443
```

```
min    1488.000000    61.000000          7.000000   13.000000    16.000000
25%    2145.000000    97.000000          8.600000   19.000000    25.000000
50%    2414.000000   120.000000          9.000000   24.000000    30.000000
75%    2935.000000   141.000000          9.400000   30.000000    34.000000
max    4066.000000   326.000000         23.000000   49.000000    54.000000
```

-> Display only ","width","height","curb-weight","engine-type" columns.

In [303…
```
print(df[["width", "height", "curb-weight", "engine-type"]])
```

```
     width  height  curb-weight engine-type
0     64.1    48.8         2548        dohc
1     64.1    48.8         2548        dohc
2     65.5    52.4         2823        ohcv
3     66.2    54.3         2337         ohc
4     66.4    54.3         2824         ohc
..     ...     ...          ...         ...
200   68.9    55.5         2952         ohc
201   68.8    55.5         3049         ohc
202   68.9    55.5         3012        ohcv
203   68.9    55.5         3217         ohc
204   68.9    55.5         3062         ohc

[205 rows x 4 columns]
```

-> Display car details which have num-of-doors = four

In [307…
```
print(df[df["num-of-doors"] == "four"])
```

```
     symboling normalized-losses   make fuel-type aspiration num-of-doors  \
3            2               164   audi       gas        std         four
4            2               164   audi       gas        std         four
6            1               158   audi       gas        std         four
7            1                 ?   audi       gas        std         four
8            1               158   audi       gas      turbo         four
..         ...               ...    ...       ...        ...          ...
200         -1                95  volvo       gas        std         four
201         -1                95  volvo       gas      turbo         four
202         -1                95  volvo       gas        std         four
203         -1                95  volvo    diesel      turbo         four
204         -1                95  volvo       gas      turbo         four

    body-style drive-wheels engine-location  wheel-base  ...  engine-size  \
3        sedan          fwd           front        99.8  ...          109
4        sedan          4wd           front        99.4  ...          136
6        sedan          fwd           front       105.8  ...          136
7        wagon          fwd           front       105.8  ...          136
8        sedan          fwd           front       105.8  ...          131
..         ...          ...             ...         ...  ...          ...
200      sedan          rwd           front       109.1  ...          141
201      sedan          rwd           front       109.1  ...          141
202      sedan          rwd           front       109.1  ...          173
203      sedan          rwd           front       109.1  ...          145
204      sedan          rwd           front       109.1  ...          141

     fuel-system  bore  stroke compression-ratio horsepower  peak-rpm  \
3           mpfi  3.19    3.40              10.0        102      5500
4           mpfi  3.19    3.40               8.0        115      5500
6           mpfi  3.19    3.40               8.5        110      5500
7           mpfi  3.19    3.40               8.5        110      5500
8           mpfi  3.13    3.40               8.3        140      5500
..           ...   ...     ...               ...        ...       ...
200         mpfi  3.78    3.15               9.5        114      5400
201         mpfi  3.78    3.15               8.7        160      5300
202         mpfi  3.58    2.87               8.8        134      5500
203          idi  3.01    3.40              23.0        106      4800
204         mpfi  3.78    3.15               9.5        114      5400

    city-mpg highway-mpg  price
3         24          30  13950
4         18          22  17450
6         19          25  17710
7         19          25  18920
8         17          20  23875
..       ...         ...    ...
200       23          28  16845
201       19          25  19045
202       18          23  21485
203       26          27  22470
204       19          25  22625

[114 rows x 26 columns]
```

3. Missing Value Handlling

-> Missing value is represented by '?' in this dataset. Replace it with NULL.

In [311…
```
df.replace('?', None , inplace=True)
print(df.head())
```

```
  symboling normalized-losses         make fuel-type aspiration num-of-doors  \
0         3              None  alfa-romero       gas        std          two
1         3              None  alfa-romero       gas        std          two
2         1              None  alfa-romero       gas        std          two
3         2               164         audi       gas        std         four
4         2               164         audi       gas        std         four

    body-style drive-wheels engine-location  wheel-base  ...  engine-size  \
0  convertible          rwd           front        88.6  ...          130
1  convertible          rwd           front        88.6  ...          130
2    hatchback          rwd           front        94.5  ...          152
3        sedan          fwd           front        99.8  ...          109
4        sedan          4wd           front        99.4  ...          136

   fuel-system  bore  stroke compression-ratio horsepower  peak-rpm city-mpg  \
0         mpfi  3.47    2.68               9.0        111      5000       21
1         mpfi  3.47    2.68               9.0        111      5000       21
2         mpfi  2.68    3.47               9.0        154      5000       19
3         mpfi  3.19    3.40              10.0        102      5500       24
4         mpfi  3.19    3.40               8.0        115      5500       18

  highway-mpg  price
0          27  13495
1          27  16500
2          26  16500
3          30  13950
4          22  17450

[5 rows x 26 columns]
```

-> Check how many missing values are there in each attribute.

In [316…
```
print(df.isnull().sum())
```

```
symboling             0
normalized-losses    41
make                  0
fuel-type             0
aspiration            0
num-of-doors          2
body-style            0
drive-wheels          0
engine-location       0
wheel-base            0
length                0
width                 0
height                0
curb-weight           0
engine-type           0
num-of-cylinders      0
engine-size           0
fuel-system           0
bore                  4
stroke                4
compression-ratio     0
horsepower            2
peak-rpm              2
city-mpg              0
highway-mpg           0
price                 4
dtype: int64
```

-> Replace missing values of "normalized-losses", "stroke", "bore", "horsepower" with mean.

In [26]:
```
for col in ["normalized-losses", "stroke", "bore", "horsepower"]:
    df[col].fillna(df[col].mean(), inplace=True)
print(df.head())
```

```
   symboling  normalized-losses         make fuel-type aspiration  \
0          3              122.0  alfa-romero       gas        std
1          3              122.0  alfa-romero       gas        std
2          1              122.0  alfa-romero       gas        std
3          2              164.0         audi       gas        std
4          2              164.0         audi       gas        std

  num-of-doors   body-style drive-wheels engine-location  wheelbase  ...  \
0          two  convertible          rwd           front       88.6  ...
1          two  convertible          rwd           front       88.6  ...
2          two    hatchback          rwd           front       94.5  ...
3         four        sedan          fwd           front       99.8  ...
4         four        sedan          4wd           front       99.4  ...

   engine-size  fuel-system  bore  stroke compression-ratio horsepower  \
0          130         mpfi  3.47    2.68               9.0      111.0
1          130         mpfi  3.47    2.68               9.0      111.0
2          152         mpfi  2.68    3.47               9.0      154.0
3          109         mpfi  3.19    3.40              10.0      102.0
4          136         mpfi  3.19    3.40               8.0      115.0

   peak-rpm city-mpg  highway-mpg    price
0    5000.0       21           27  13495.0
1    5000.0       21           27  16500.0
2    5000.0       19           26  16500.0
3    5500.0       24           30  13950.0
4    5500.0       18           22  17450.0

[5 rows x 26 columns]
```

```
C:\Users\rajva\AppData\Local\Temp\ipykernel_19968\497043668.py:2: FutureWarning: A v
alue is trying to be set on a copy of a DataFrame or Series through chained assignme
nt using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df[col].fillna(df[col].mean(), inplace=True)
```

-> Drop all the rows which has missing value in attribute “price”.

In [31]:
```
df.dropna(subset=["price"], inplace=True)
print(df.head())
```

```
   symboling  normalized-losses         make fuel-type aspiration  \
0          3              122.0  alfa-romero       gas        std
1          3              122.0  alfa-romero       gas        std
2          1              122.0  alfa-romero       gas        std
3          2              164.0         audi       gas        std
4          2              164.0         audi       gas        std

  num-of-doors   body-style drive-wheels engine-location  wheelbase  ...  \
0          two  convertible          rwd           front       88.6  ...
1          two  convertible          rwd           front       88.6  ...
2          two    hatchback          rwd           front       94.5  ...
3         four        sedan          fwd           front       99.8  ...
4         four        sedan          4wd           front       99.4  ...

   engine-size  fuel-system  bore  stroke compression-ratio horsepower  \
0          130         mpfi  3.47    2.68               9.0      111.0
1          130         mpfi  3.47    2.68               9.0      111.0
2          152         mpfi  2.68    3.47               9.0      154.0
3          109         mpfi  3.19    3.40              10.0      102.0
4          136         mpfi  3.19    3.40               8.0      115.0

   peak-rpm city-mpg  highway-mpg    price
0    5000.0       21           27  13495.0
1    5000.0       21           27  16500.0
2    5000.0       19           26  16500.0
3    5500.0       24           30  13950.0
4    5500.0       18           22  17450.0

[5 rows x 26 columns]
```

-> Replace missing values of “num-of-doors” with mode.

In [41]:
```
df["num-of-doors"].fillna(df["num-of-doors"].mode()[0], inplace=True)
print(df.head())
```

```
   symboling  normalized-losses         make fuel-type aspiration  \
0          3              122.0  alfa-romero       gas        std
1          3              122.0  alfa-romero       gas        std
2          1              122.0  alfa-romero       gas        std
3          2              164.0         audi       gas        std
4          2              164.0         audi       gas        std

  num-of-doors   body-style drive-wheels engine-location  wheelbase  ...  \
0          two  convertible          rwd           front       88.6  ...
1          two  convertible          rwd           front       88.6  ...
2          two    hatchback          rwd           front       94.5  ...
3         four        sedan          fwd           front       99.8  ...
4         four        sedan          4wd           front       99.4  ...

   engine-size  fuel-system  bore  stroke compression-ratio horsepower  \
0          130         mpfi  3.47    2.68               9.0      111.0
1          130         mpfi  3.47    2.68               9.0      111.0
2          152         mpfi  2.68    3.47               9.0      154.0
3          109         mpfi  3.19    3.40              10.0      102.0
4          136         mpfi  3.19    3.40               8.0      115.0

   peak-rpm city-mpg  highway-mpg    price
0    5000.0       21           27  13495.0
1    5000.0       21           27  16500.0
2    5000.0       19           26  16500.0
3    5500.0       24           30  13950.0
4    5500.0       18           22  17450.0

[5 rows x 26 columns]
```

```
C:\Users\rajva\AppData\Local\Temp\ipykernel_19968\2149171378.py:1: FutureWarning: A
value is trying to be set on a copy of a DataFrame or Series through chained assignm
ent using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df["num-of-doors"].fillna(df["num-of-doors"].mode()[0], inplace=True)
```

-> Replace all other missing values with median.

In [39]:
```
for col in df.columns:
    if df[col].isnull().any():
        df[col] = df[col].fillna(df[col].median())
print(df.head())
```

```
   symboling  normalized-losses         make fuel-type aspiration  \
0          3              122.0  alfa-romero       gas        std
1          3              122.0  alfa-romero       gas        std
2          1              122.0  alfa-romero       gas        std
3          2              164.0         audi       gas        std
4          2              164.0         audi       gas        std

  num-of-doors   body-style drive-wheels engine-location  wheelbase  ...  \
0          two  convertible          rwd           front       88.6  ...
1          two  convertible          rwd           front       88.6  ...
2          two    hatchback          rwd           front       94.5  ...
3         four        sedan          fwd           front       99.8  ...
4         four        sedan          4wd           front       99.4  ...

   engine-size  fuel-system  bore  stroke compression-ratio horsepower  \
0          130         mpfi  3.47    2.68               9.0      111.0
1          130         mpfi  3.47    2.68               9.0      111.0
2          152         mpfi  2.68    3.47               9.0      154.0
3          109         mpfi  3.19    3.40              10.0      102.0
4          136         mpfi  3.19    3.40               8.0      115.0

   peak-rpm city-mpg  highway-mpg    price
0    5000.0       21           27  13495.0
1    5000.0       21           27  16500.0
2    5000.0       19           26  16500.0
3    5500.0       24           30  13950.0
4    5500.0       18           22  17450.0

[5 rows x 26 columns]
```

4.Grouping, Sorting, and Aggregation

-> Group by “Fuel-type” and compute average price.

In [48]:
```
df["price"] = pd.to_numeric(df["price"], errors='coerce')
avg_price_by_fuel = df.groupby("fuel-type")["price"].mean()
print("Average price by fuel-type:\n", avg_price_by_fuel)
```

```
Average price by fuel-type:
 fuel-type
diesel    15838.15000
gas       12916.40884
Name: price, dtype: float64
```

-> In our dataset, the fuel consumption columns "city-mpg" and "highway-mpg" are represented by mpg (miles per gallon) unit. Assume we are developing an application in a country that accept the fuel consumption with L/100km standard We will need to apply data transformation to transform mpg into L/100km? The formula for unit conversion is L/100km = 235 / mpg

In [53]:
```
df["city-L/100km"] = 235 / df["city-mpg"]
df["highway-L/100km"] = 235 / df["highway-mpg"]
print(df[["city-mpg", "city-L/100km", "highway-mpg", "highway-L/100km"]].head())
```

```
   city-mpg  city-L/100km  highway-mpg  highway-L/100km
0        21     11.190476           27         8.703704
1        21     11.190476           27         8.703704
2        19     12.368421           26         9.038462
3        24      9.791667           30         7.833333
4        18     13.055556           22        10.681818
```

-> Sort the DataFrame based on “price” in descending order.

In [57]:
```
df_sorted = df.sort_values(by="price", ascending=False)
print(df_sorted[["make", "price"]].head())
```

```
              make    price
74   mercedes-benz  45400.0
16             bmw  41315.0
73   mercedes-benz  40960.0
128        porsche  37028.0
17             bmw  36880.0
```

In [ ]:

NumPy

1. NumPy Basic Ops

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

-> Create a 1D Array from 1 to 20.

In [186…
```
array_1d = np.arange(1, 21)
print("1D Array from 1 to 20:\n", array_1d)
```

```
1D Array from 1 to 20:
 [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20]
```

-> Create a 3×4 matrix of ones and reshape it to 4×3

In [189…
```
mat_1 = np.ones((3,4))
print("\n 3x4 Matrix of Ones: \n", mat_1)

# Reshapping It to 4x3

reshape = mat_1.reshape((4,3))
print("\n 4x3 Reshaped Materix of Ones: \n", reshape)
```

```
3x4 Matrix of Ones:
[[1. 1. 1. 1.]
[1. 1. 1. 1.]
[1. 1. 1. 1.]]

4x3 Reshaped Materix of Ones:
[[1. 1. 1.]
[1. 1. 1.]
[1. 1. 1.]
[1. 1. 1.]]
```

-> Create a 5×5 identity matrix

In [194…
```
i5 = np.eye(5)
print("\n 5x5 Identity Matrix: \n", i5)
```

```
5x5 Identity Matrix:
[[1. 0. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 0. 1. 0. 0.]
[0. 0. 0. 1. 0.]
[0. 0. 0. 0. 1.]]
```

-> Generate 15 equally spaced numbers between 5 and 50.

In [199…
```
e_nums = np.linspace(5,50,15)
print("\n 15 equally spaced nums between 5 and 15: \n", e_nums)
```

```
15 equally spaced nums between 5 and 15:
[ 5.          8.21428571 11.42857143 14.64285714 17.85714286 21.07142857
24.28571429 27.5        30.71428571 33.92857143 37.14285714 40.35714286
43.57142857 46.78571429 50.        ]
```

-> Generate a 4×4 matrix of random integers between 1 and 100.

In [201…
```
r_nums = np.random.randint(1,101,size=(4,4))
print("\n 4x4 matrix of Random int between 1 & 100 \n",r_nums)
```

```
4x4 matrix of Random int between 1 & 100
[[48 17 75 76]
[47 94 95 48]
[36 68 89 44]
[93 58 61 56]]
```

2. NumPy Array Ops

-> Create a 3×3 matrix of random integers between 1 and 100

In [205…
```
r_mat = np.random.randint(1,100,size=(3,3))
print("\n 3x3 Matrix of random nums \n",r_mat)
```

```
3x3 Matrix of random nums
[[34 34 99]
[14 14 29]
[37 85  1]]
```

-> Extract: First row, Second column, Center element.

In [208…
```
fr = r_mat[0]
print("\n First Row: \n",fr)

sc = r_mat[:,1]
print("\n Second Column \n",sc)

se = r_mat[1,1]
print("\n Center Element \n",se)
```

```
First Row:
[34 34 99]

Second Column
[34 14 85]

Center Element
14
```

-> Replace all values greater than 50 in a matrix with 999.

In [211…
```
r_mat_c = r_mat.copy()
r_mat_c[r_mat_c > 50] = 999
print("\n Replaced Matrix \n",r_mat_c)
```

```
Replaced Matrix
[[ 34  34 999]
[ 14  14  29]
[ 37 999   1]]
```

-> Multiply a 1D array of size 5 (random integers between 1 to 10) by 10 using broadcasting.

In [214…
```
arr_1d = np.random.randint(1,11,size=5)
multi_arr = arr_1d * 10
print("\n Original 1D Array: \n",arr_1d)
print("\n Multiplied Array: \n",multi_arr)
```

```
Original 1D Array:
[10  9  3  8  6]

Multiplied Array:
[100  90  30  80  60]
```

3. Math & Stats Ops

->Create a 3×3 matrix of random integers between 1 and 100

In [225…
```
r_mat = np.random.randint(1,100,size=(3,3))
print("\n 3x3 Matrix of random nums \n",r_mat)
```

```
3x3 Matrix of random nums
[[24  1 95]
[79 17 61]
[99 93 46]]
```

-> Compute sum, mean, median, std, var, min, and max of the above array.

In [230…
```
print("\nStatistics:")
print("Sum:", np.sum(r_mat))
print("Mean:", np.mean(r_mat))
```

```
print("Median:", np.median(r_mat))
print("Standard Deviation:", np.std(r_mat))
print("Variance:", np.var(r_mat))
print("Minimum:", np.min(r_mat))
print("Maximum:", np.max(r_mat))
```

```
Statistics:
Sum: 515
Mean: 57.22222222222222
Median: 61.0
Standard Deviation: 34.880041341010724
Variance: 1216.6172839506173
Minimum: 1
Maximum: 99
```

-> Normalize a 1D array of size 5 (random integers between 1 to 10) to scale values between 0 and 1.

In [234…

```
arr_1d = np.random.randint(1,11,size=5)
print("\n 1D Array: \n",arr_1d)
min = np.min(arr_1d)
max = np.max(arr_1d)
norm_arr = ((arr_1d - min) / (max - min) )
print("\n Normalized Array: \n",norm_arr)
```

```
1D Array:
[ 8 10  5  1  4]

Normalized Array:
[0.77777778 1.         0.44444444 0.         0.33333333]
```

4. NumPy Matrix Ops & Algebra

-> Generate Following two NumPy matrix import numpy as np

A = ([4, 2], [1, 3])

B = ([2, 0], [1, 5])

In [240…

```
A = np.array( [[4, 2], [1, 3]] )
B = np.array( [[2, 0], [1, 5]] )

print("\n A: \n",A)
print("\n B: \n",B)
```

```
A:
[[4 2]
[1 3]]

B:
[[2 0]
[1 5]]
```

-> Matrix Multiplication

In [252…
```
mat_multi = np.matmul(A,B)
print("\n Multiplication of A & B: \n",mat_multi)
```

```
Multiplication of A & B:
[[10 10]
[ 5 15]]
```

-> dot product of A and B.

In [254…
```
dot = np.dot(A,B)
print("\n dot product of A & B.: \n",dot)
```

```
dot product of A & B.:
[[10 10]
[ 5 15]]
```

-> Element wise addition/ subtraction/ multiplications/ division of A and B

In [250…
```
add = A + B
sub = A - B
multi = A * B
div = A / B

print("\nElement-wise Addition:\n", add)
print("Element-wise Subtraction:\n", sub)
print("Element-wise Multiplication:\n", multi)
print("Element-wise Division:\n", div)
```

```
Element-wise Addition:
 [[6 2]
 [2 8]]
Element-wise Subtraction:
 [[ 2  2]
 [ 0 -2]]
Element-wise Multiplication:
 [[ 8  0]
 [ 1 15]]
Element-wise Division:
 [[2.  inf]
 [1.  0.6]]
C:\Users\rajva\AppData\Local\Temp\ipykernel_18728\3905969198.py:4: RuntimeWarning: d
ivide by zero encountered in divide
  div = A / B
```

-> Transpose of A

In [277…
```
trans_A = A.T
print("\n Transpose of A: \n",trans_A)
```

```
Transpose of A:
[[4 1]
[2 3]]
```

-> Determinant of A

In [271…
```
det_A = np.linalg.det(A)
print("\n Determinant of A: \n",det_A)
```

```
Determinant of A:
10.000000000000002
```

-> Inverse of A

In [269…
```
if det_A != 0:
    inv_A = np.linalg.inv(A)
    print("\nInverse of A:\n", inv_A)
else:
    print("\nMatrix A cannot be inverted.")
```

```
Inverse of A:
 [[ 0.3 -0.2]
 [-0.1  0.4]]
```

-> Eigenvalues and Eigenvectors of A

In [264…
```
eigenvalues= np.linalg.eig(A)
eigenvectors = np.linalg.eig(A)
print("\nEigenvalues of A:\n", eigenvalues)
print("\nEigenvectors of A:\n", eigenvectors)
```

```
Eigenvalues of A:
 EigResult(eigenvalues=array([5., 2.]), eigenvectors=array([[ 0.89442719, -0.7071067
8],
       [ 0.4472136 ,  0.70710678]]))

Eigenvectors of A:
 EigResult(eigenvalues=array([5., 2.]), eigenvectors=array([[ 0.89442719, -0.7071067
8],
       [ 0.4472136 ,  0.70710678]]))
```

-> Solve System of Equations:

2x + y = 8

3x + 4y = 18

In [261…
```
coeff = np.array([[2,1],[3,4]])
cons = np.array([8,18])

sol = np.linalg.solve(coeff, cons)
print("\n Solution of the Equations x & y : ",sol)
```

```
Solution of the Equations x & y :  [2.8 2.4]
```

Pandas Exercise

1. Series & DataFrame Basics

-> Given the following list of marks:

[78, 85, 92, 70, 66]

Create a Pandas Series and assign the following student names as indices:

['Amit', 'Bhavna', 'Chetan', 'Divya', 'Esha']

Display the Series

In [ ]:
```
marks = [78, 85, 92, 70, 66]
names = ['Amit', 'Bhavna', 'Chetan', 'Divya', 'Esha']

marks_series = pd.Series(data=marks,index=names)
print("\nPandas Series : \n",marks_series)
```

-> Create a Pandas DataFrame and display:

The full DataFrame

The column names

The shape of the DataFrame

In [285…
```
# Create DataFrame
data = {
    'Name': ['Amit', 'Bhavna', 'Chetan', 'Divya', 'Esha'],
    'Gender': ['Male', 'Female', 'Male', 'Female', 'Female'],
    'Math': [78, 85, 92, 70, 66],
    'Science': [88, 79, 95, 72, 60]
}

df = pd.DataFrame(data)

print("\nFull DataFrame:\n")
print(df)
```

```
# Display column names
print("\nColumn Names:\n")
print(df.columns.tolist())

# Display shape (rows, columns)
print("\nShape of the DataFrame:\n")
print(df.shape)
```

```
Full DataFrame:

     Name  Gender  Math  Science
0    Amit    Male    78       88
1  Bhavna  Female    85       79
2  Chetan    Male    92       95
3   Divya  Female    70       72
4    Esha  Female    66       60

Column Names:

['Name', 'Gender', 'Math', 'Science']

Shape of the DataFrame:

(5, 4)
```

2.Data Exploration

-> Load the dataset from this url "https://archive.ics.uci.edu/ml/machine-learningdatabases/autos/imports-85.data"

In [7]:
```
import pandas as pd
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.d
df = pd.read_csv(url, header=None)
print(df.head())
```

```
   0    1            2    3    4     5            6    7      8     9   ...  \
0   3    ?  alfa-romero  gas  std   two  convertible  rwd  front  88.6  ...
1   3    ?  alfa-romero  gas  std   two  convertible  rwd  front  88.6  ...
2   1    ?  alfa-romero  gas  std   two    hatchback  rwd  front  94.5  ...
3   2  164         audi  gas  std  four        sedan  fwd  front  99.8  ...
4   2  164         audi  gas  std  four        sedan  4wd  front  99.4  ...

    16    17    18    19    20   21    22  23  24     25
0  130  mpfi  3.47  2.68   9.0  111  5000  21  27  13495
1  130  mpfi  3.47  2.68   9.0  111  5000  21  27  16500
2  152  mpfi  2.68  3.47   9.0  154  5000  19  26  16500
3  109  mpfi  3.19  3.40  10.0  102  5500  24  30  13950
4  136  mpfi  3.19  3.40   8.0  115  5500  18  22  17450

[5 rows x 26 columns]
```

-> Assign names of columns:

["symboling","normalized-losses","make","fuel-type","aspiration", "num-ofdoors","body-style", "drive-wheels","engine-ocation","wheelbase","length","width","height","curb-weight","engine-type","num-of-cylinders","engine-size","fuel-system","bore","stroke","compression-ratio","horsepower","peak-rpm","city-mpg","highway-mpg","price"]

In [296…
```
df.columns = ["symboling","normalized-losses","make","fuel-type","aspiration", "num
print(df.head())
```

```
  symboling normalized-losses         make fuel-type aspiration num-of-doors  \
0         3                 ?  alfa-romero       gas        std          two
1         3                 ?  alfa-romero       gas        std          two
2         1                 ?  alfa-romero       gas        std          two
3         2               164         audi       gas        std         four
4         2               164         audi       gas        std         four

    body-style drive-wheels engine-location  wheel-base  ...  engine-size  \
0  convertible          rwd           front        88.6  ...          130
1  convertible          rwd           front        88.6  ...          130
2    hatchback          rwd           front        94.5  ...          152
3        sedan          fwd           front        99.8  ...          109
4        sedan          4wd           front        99.4  ...          136

   fuel-system  bore  stroke compression-ratio horsepower  peak-rpm city-mpg  \
0         mpfi  3.47    2.68               9.0        111      5000       21
1         mpfi  3.47    2.68               9.0        111      5000       21
2         mpfi  2.68    3.47               9.0        154      5000       19
3         mpfi  3.19    3.40              10.0        102      5500       24
4         mpfi  3.19    3.40               8.0        115      5500       18

  highway-mpg  price
0          27  13495
1          27  16500
2          26  16500
3          30  13950
4          22  17450

[5 rows x 26 columns]
```

-> Display .shape, .columns, .info(), and .describe()

In [300…
```
print(df.shape)
print(df.columns)
print(df.info())
print(df.describe())
```

```
(205, 26)
Index(['symboling', 'normalized-losses', 'make', 'fuel-type', 'aspiration',
       'num-of-doors', 'body-style', 'drive-wheels', 'engine-location',
       'wheel-base', 'length', 'width', 'height', 'curb-weight', 'engine-type',
       'num-of-cylinders', 'engine-size', 'fuel-system', 'bore', 'stroke',
       'compression-ratio', 'horsepower', 'peak-rpm', 'city-mpg',
       'highway-mpg', 'price'],
      dtype='object')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   symboling          205 non-null    int64
 1   normalized-losses  205 non-null    object
 2   make               205 non-null    object
 3   fuel-type          205 non-null    object
 4   aspiration         205 non-null    object
 5   num-of-doors       205 non-null    object
 6   body-style         205 non-null    object
 7   drive-wheels       205 non-null    object
 8   engine-location    205 non-null    object
 9   wheel-base         205 non-null    float64
 10  length             205 non-null    float64
 11  width              205 non-null    float64
 12  height             205 non-null    float64
 13  curb-weight        205 non-null    int64
 14  engine-type        205 non-null    object
 15  num-of-cylinders   205 non-null    object
 16  engine-size        205 non-null    int64
 17  fuel-system        205 non-null    object
 18  bore               205 non-null    object
 19  stroke             205 non-null    object
 20  compression-ratio  205 non-null    float64
 21  horsepower         205 non-null    object
 22  peak-rpm           205 non-null    object
 23  city-mpg           205 non-null    int64
 24  highway-mpg        205 non-null    int64
 25  price              205 non-null    object
dtypes: float64(5), int64(5), object(16)
memory usage: 41.8+ KB
None
        symboling  wheel-base      length       width      height  \
count  205.000000  205.000000  205.000000  205.000000  205.000000
mean     0.834146   98.756585  174.049268   65.907805   53.724878
std      1.245307    6.021776   12.337289    2.145204    2.443522
min     -2.000000   86.600000  141.100000   60.300000   47.800000
25%      0.000000   94.500000  166.300000   64.100000   52.000000
50%      1.000000   97.000000  173.200000   65.500000   54.100000
75%      2.000000  102.400000  183.100000   66.900000   55.500000
max      3.000000  120.900000  208.100000   72.300000   59.800000

       curb-weight  engine-size  compression-ratio    city-mpg  highway-mpg
count   205.000000   205.000000         205.000000  205.000000   205.000000
mean   2555.565854   126.907317          10.142537   25.219512    30.751220
std     520.680204    41.642693           3.972040    6.542142     6.886443
```

```
min    1488.000000    61.000000          7.000000   13.000000    16.000000
25%    2145.000000    97.000000          8.600000   19.000000    25.000000
50%    2414.000000   120.000000          9.000000   24.000000    30.000000
75%    2935.000000   141.000000          9.400000   30.000000    34.000000
max    4066.000000   326.000000         23.000000   49.000000    54.000000
```

-> Display only ","width","height","curb-weight","engine-type" columns.

In [303…
```
print(df[["width", "height", "curb-weight", "engine-type"]])
```

```
     width  height  curb-weight engine-type
0     64.1    48.8         2548        dohc
1     64.1    48.8         2548        dohc
2     65.5    52.4         2823        ohcv
3     66.2    54.3         2337         ohc
4     66.4    54.3         2824         ohc
..     ...     ...          ...         ...
200   68.9    55.5         2952         ohc
201   68.8    55.5         3049         ohc
202   68.9    55.5         3012        ohcv
203   68.9    55.5         3217         ohc
204   68.9    55.5         3062         ohc

[205 rows x 4 columns]
```

-> Display car details which have num-of-doors = four

In [307…
```
print(df[df["num-of-doors"] == "four"])
```

```
     symboling normalized-losses   make fuel-type aspiration num-of-doors  \
3            2               164   audi       gas        std         four
4            2               164   audi       gas        std         four
6            1               158   audi       gas        std         four
7            1                 ?   audi       gas        std         four
8            1               158   audi       gas      turbo         four
..         ...               ...    ...       ...        ...          ...
200         -1                95  volvo       gas        std         four
201         -1                95  volvo       gas      turbo         four
202         -1                95  volvo       gas        std         four
203         -1                95  volvo    diesel      turbo         four
204         -1                95  volvo       gas      turbo         four

    body-style drive-wheels engine-location  wheel-base  ...  engine-size  \
3        sedan          fwd           front        99.8  ...          109
4        sedan          4wd           front        99.4  ...          136
6        sedan          fwd           front       105.8  ...          136
7        wagon          fwd           front       105.8  ...          136
8        sedan          fwd           front       105.8  ...          131
..         ...          ...             ...         ...  ...          ...
200      sedan          rwd           front       109.1  ...          141
201      sedan          rwd           front       109.1  ...          141
202      sedan          rwd           front       109.1  ...          173
203      sedan          rwd           front       109.1  ...          145
204      sedan          rwd           front       109.1  ...          141

     fuel-system  bore  stroke compression-ratio horsepower  peak-rpm  \
3           mpfi  3.19    3.40              10.0        102      5500
4           mpfi  3.19    3.40               8.0        115      5500
6           mpfi  3.19    3.40               8.5        110      5500
7           mpfi  3.19    3.40               8.5        110      5500
8           mpfi  3.13    3.40               8.3        140      5500
..           ...   ...     ...               ...        ...       ...
200         mpfi  3.78    3.15               9.5        114      5400
201         mpfi  3.78    3.15               8.7        160      5300
202         mpfi  3.58    2.87               8.8        134      5500
203          idi  3.01    3.40              23.0        106      4800
204         mpfi  3.78    3.15               9.5        114      5400

    city-mpg highway-mpg  price
3         24          30  13950
4         18          22  17450
6         19          25  17710
7         19          25  18920
8         17          20  23875
..       ...         ...    ...
200       23          28  16845
201       19          25  19045
202       18          23  21485
203       26          27  22470
204       19          25  22625

[114 rows x 26 columns]
```

3. Missing Value Handlling

-> Missing value is represented by '?' in this dataset. Replace it with NULL.

In [311…
```
df.replace('?', None , inplace=True)
print(df.head())
```

```
   symboling normalized-losses         make fuel-type aspiration num-of-doors  \
0          3              None  alfa-romero       gas        std          two
1          3              None  alfa-romero       gas        std          two
2          1              None  alfa-romero       gas        std          two
3          2               164         audi       gas        std         four
4          2               164         audi       gas        std         four

    body-style drive-wheels engine-location  wheel-base  ...  engine-size  \
0  convertible          rwd           front        88.6  ...          130
1  convertible          rwd           front        88.6  ...          130
2    hatchback          rwd           front        94.5  ...          152
3        sedan          fwd           front        99.8  ...          109
4        sedan          4wd           front        99.4  ...          136

   fuel-system  bore  stroke compression-ratio horsepower  peak-rpm city-mpg  \
0         mpfi  3.47    2.68               9.0        111      5000       21
1         mpfi  3.47    2.68               9.0        111      5000       21
2         mpfi  2.68    3.47               9.0        154      5000       19
3         mpfi  3.19    3.40              10.0        102      5500       24
4         mpfi  3.19    3.40               8.0        115      5500       18

  highway-mpg  price
0          27  13495
1          27  16500
2          26  16500
3          30  13950
4          22  17450

[5 rows x 26 columns]
```

-> Check how many missing values are there in each attribute.

In [316…
```
print(df.isnull().sum())
```

```
symboling             0
normalized-losses    41
make                  0
fuel-type             0
aspiration            0
num-of-doors          2
body-style            0
drive-wheels          0
engine-location       0
wheel-base            0
length                0
width                 0
height                0
curb-weight           0
engine-type           0
num-of-cylinders      0
engine-size           0
fuel-system           0
bore                  4
stroke                4
compression-ratio     0
horsepower            2
peak-rpm              2
city-mpg              0
highway-mpg           0
price                 4
dtype: int64
```

-> Replace missing values of "normalized-losses", "stroke", "bore", "horsepower" with mean.

In [26]:
```
for col in ["normalized-losses", "stroke", "bore", "horsepower"]:
    df[col].fillna(df[col].mean(), inplace=True)
print(df.head())
```

```
   symboling  normalized-losses         make fuel-type aspiration  \
0          3              122.0  alfa-romero       gas        std
1          3              122.0  alfa-romero       gas        std
2          1              122.0  alfa-romero       gas        std
3          2              164.0         audi       gas        std
4          2              164.0         audi       gas        std

  num-of-doors   body-style drive-wheels engine-location  wheelbase  ...  \
0          two  convertible          rwd           front       88.6  ...
1          two  convertible          rwd           front       88.6  ...
2          two    hatchback          rwd           front       94.5  ...
3         four        sedan          fwd           front       99.8  ...
4         four        sedan          4wd           front       99.4  ...

   engine-size  fuel-system  bore  stroke compression-ratio horsepower  \
0          130         mpfi  3.47    2.68               9.0      111.0
1          130         mpfi  3.47    2.68               9.0      111.0
2          152         mpfi  2.68    3.47               9.0      154.0
3          109         mpfi  3.19    3.40              10.0      102.0
4          136         mpfi  3.19    3.40               8.0      115.0

   peak-rpm city-mpg  highway-mpg    price
0    5000.0       21           27  13495.0
1    5000.0       21           27  16500.0
2    5000.0       19           26  16500.0
3    5500.0       24           30  13950.0
4    5500.0       18           22  17450.0

[5 rows x 26 columns]
```

```
C:\Users\rajva\AppData\Local\Temp\ipykernel_19968\497043668.py:2: FutureWarning: A v
alue is trying to be set on a copy of a DataFrame or Series through chained assignme
nt using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df[col].fillna(df[col].mean(), inplace=True)
```

-> Drop all the rows which has missing value in attribute “price”.

In [31]:
```
df.dropna(subset=["price"], inplace=True)
print(df.head())
```

```
   symboling  normalized-losses         make fuel-type aspiration  \
0          3              122.0  alfa-romero       gas        std
1          3              122.0  alfa-romero       gas        std
2          1              122.0  alfa-romero       gas        std
3          2              164.0         audi       gas        std
4          2              164.0         audi       gas        std

  num-of-doors   body-style drive-wheels engine-location  wheelbase  ...  \
0          two  convertible          rwd           front       88.6  ...
1          two  convertible          rwd           front       88.6  ...
2          two    hatchback          rwd           front       94.5  ...
3         four        sedan          fwd           front       99.8  ...
4         four        sedan          4wd           front       99.4  ...

   engine-size  fuel-system  bore  stroke compression-ratio horsepower  \
0          130         mpfi  3.47    2.68               9.0      111.0
1          130         mpfi  3.47    2.68               9.0      111.0
2          152         mpfi  2.68    3.47               9.0      154.0
3          109         mpfi  3.19    3.40              10.0      102.0
4          136         mpfi  3.19    3.40               8.0      115.0

   peak-rpm city-mpg  highway-mpg    price
0    5000.0       21           27  13495.0
1    5000.0       21           27  16500.0
2    5000.0       19           26  16500.0
3    5500.0       24           30  13950.0
4    5500.0       18           22  17450.0

[5 rows x 26 columns]
```

-> Replace missing values of “num-of-doors” with mode.

In [41]:
```
df["num-of-doors"].fillna(df["num-of-doors"].mode()[0], inplace=True)
print(df.head())
```

```
   symboling  normalized-losses         make fuel-type aspiration  \
0          3              122.0  alfa-romero       gas        std
1          3              122.0  alfa-romero       gas        std
2          1              122.0  alfa-romero       gas        std
3          2              164.0         audi       gas        std
4          2              164.0         audi       gas        std

  num-of-doors   body-style drive-wheels engine-location  wheelbase  ...  \
0          two  convertible          rwd           front       88.6  ...
1          two  convertible          rwd           front       88.6  ...
2          two    hatchback          rwd           front       94.5  ...
3         four        sedan          fwd           front       99.8  ...
4         four        sedan          4wd           front       99.4  ...

   engine-size  fuel-system  bore  stroke compression-ratio horsepower  \
0          130         mpfi  3.47    2.68               9.0      111.0
1          130         mpfi  3.47    2.68               9.0      111.0
2          152         mpfi  2.68    3.47               9.0      154.0
3          109         mpfi  3.19    3.40              10.0      102.0
4          136         mpfi  3.19    3.40               8.0      115.0

   peak-rpm city-mpg  highway-mpg    price
0    5000.0       21           27  13495.0
1    5000.0       21           27  16500.0
2    5000.0       19           26  16500.0
3    5500.0       24           30  13950.0
4    5500.0       18           22  17450.0

[5 rows x 26 columns]
```

```
C:\Users\rajva\AppData\Local\Temp\ipykernel_19968\2149171378.py:1: FutureWarning: A
value is trying to be set on a copy of a DataFrame or Series through chained assignm
ent using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df["num-of-doors"].fillna(df["num-of-doors"].mode()[0], inplace=True)
```

-> Replace all other missing values with median.

In [39]:
```
for col in df.columns:
    if df[col].isnull().any():
        df[col] = df[col].fillna(df[col].median())
print(df.head())
```

```
   symboling  normalized-losses         make fuel-type aspiration  \
0          3              122.0  alfa-romero       gas        std
1          3              122.0  alfa-romero       gas        std
2          1              122.0  alfa-romero       gas        std
3          2              164.0         audi       gas        std
4          2              164.0         audi       gas        std

  num-of-doors   body-style drive-wheels engine-location  wheelbase  ...  \
0          two  convertible          rwd           front       88.6  ...
1          two  convertible          rwd           front       88.6  ...
2          two    hatchback          rwd           front       94.5  ...
3         four        sedan          fwd           front       99.8  ...
4         four        sedan          4wd           front       99.4  ...

   engine-size  fuel-system  bore  stroke compression-ratio horsepower  \
0          130         mpfi  3.47    2.68               9.0      111.0
1          130         mpfi  3.47    2.68               9.0      111.0
2          152         mpfi  2.68    3.47               9.0      154.0
3          109         mpfi  3.19    3.40              10.0      102.0
4          136         mpfi  3.19    3.40               8.0      115.0

   peak-rpm city-mpg  highway-mpg    price
0    5000.0       21           27  13495.0
1    5000.0       21           27  16500.0
2    5000.0       19           26  16500.0
3    5500.0       24           30  13950.0
4    5500.0       18           22  17450.0

[5 rows x 26 columns]
```

4.Grouping, Sorting, and Aggregation

-> Group by “Fuel-type” and compute average price.

In [48]:
```
df["price"] = pd.to_numeric(df["price"], errors='coerce')
avg_price_by_fuel = df.groupby("fuel-type")["price"].mean()
print("Average price by fuel-type:\n", avg_price_by_fuel)
```

```
Average price by fuel-type:
 fuel-type
diesel    15838.15000
gas       12916.40884
Name: price, dtype: float64
```

-> In our dataset, the fuel consumption columns "city-mpg" and "highway-mpg" are represented by mpg (miles per gallon) unit. Assume we are developing an application in a country that accept the fuel consumption with L/100km standard We will need to apply data transformation to transform mpg into L/100km? The formula for unit conversion is L/100km = 235 / mpg

In [53]:
```
df["city-L/100km"] = 235 / df["city-mpg"]
df["highway-L/100km"] = 235 / df["highway-mpg"]
print(df[["city-mpg", "city-L/100km", "highway-mpg", "highway-L/100km"]].head())
```

```
   city-mpg  city-L/100km  highway-mpg  highway-L/100km
0        21     11.190476           27         8.703704
1        21     11.190476           27         8.703704
2        19     12.368421           26         9.038462
3        24      9.791667           30         7.833333
4        18     13.055556           22        10.681818
```

-> Sort the DataFrame based on “price” in descending order.

In [57]:
```
df_sorted = df.sort_values(by="price", ascending=False)
print(df_sorted[["make", "price"]].head())
```

```
              make    price
74   mercedes-benz  45400.0
16             bmw  41315.0
73   mercedes-benz  40960.0
128        porsche  37028.0
17             bmw  36880.0
```

In [ ]: