In [1]:
```
import pandas as pd
```

Importing Data From Local Drive

In [2]:
```
# df=pd.read_csv("imports-85.data")
```

Importing Data From Web Storage

In [3]:
```
path="https://archive.ics.uci.edu/ml/machine-learning-databases/autos/impor
```

In [4]:
```
df=pd.read_csv(path)
```

In [5]:
```
df=pd.read_csv(path,header=None)
```

In [6]:
```
df
```

Out[6]:

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.(
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.(
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.4
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.4
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
200	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	173	mpfi	3.58	2.3
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1	...	145	idi	3.01	3.4
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.

205 rows × 26 columns

In [7]:
```
df.head()
```

Out[7]:

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0

5 rows × 26 columns

In [8]:
```
df[0:10]
```

Out[8]:

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40
5	2	?	audi	gas	std	two	sedan	fwd	front	99.8	...	136	mpfi	3.19	3.40
6	1	158	audi	gas	std	four	sedan	fwd	front	105.8	...	136	mpfi	3.19	3.40
7	1	?	audi	gas	std	four	wagon	fwd	front	105.8	...	136	mpfi	3.19	3.40
8	1	158	audi	gas	turbo	four	sedan	fwd	front	105.8	...	131	mpfi	3.13	3.40
9	0	?	audi	gas	turbo	two	hatchback	4wd	front	99.5	...	131	mpfi	3.13	3.40

10 rows × 26 columns

In [9]:
```
df.tail()
```

Out[9]:

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20
200	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	8.7
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	173	mpfi	3.58	2.87	8.8
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1	...	145	idi	3.01	3.40	23.0
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5

5 rows × 26 columns

In [10]:
```
df.shape
```

Out[10]:
```
(205, 26)
```

In [11]:
```
df.dtypes
```

Out[11]:
```
0       int64
1      object
2      object
3      object
4      object
5      object
6      object
7      object
8      object
9     float64
10    float64
11    float64
12    float64
13      int64
14     object
15     object
16      int64
17     object
18     object
19     object
20    float64
21     object
22     object
23      int64
24      int64
25     object
dtype: object
```

In [12]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   0       205 non-null    int64  
 1   1       205 non-null    object 
 2   2       205 non-null    object 
 3   3       205 non-null    object 
 4   4       205 non-null    object 
 5   5       205 non-null    object 
 6   6       205 non-null    object 
 7   7       205 non-null    object 
 8   8       205 non-null    object 
 9   9       205 non-null    float64
 10  10      205 non-null    float64
 11  11      205 non-null    float64
 12  12      205 non-null    float64
 13  13      205 non-null    int64  
 14  14      205 non-null    object 
 15  15      205 non-null    object 
 16  16      205 non-null    int64  
 17  17      205 non-null    object 
 18  18      205 non-null    object 
 19  19      205 non-null    object 
 20  20      205 non-null    float64
 21  21      205 non-null    object 
 22  22      205 non-null    object 
 23  23      205 non-null    int64  
 24  24      205 non-null    int64  
 25  25      205 non-null    object 
dtypes: float64(5), int64(5), object(16)
memory usage: 41.8+ KB
```

Change Column Names

In [13]:
```
df.columns
```

Out[13]:
```
Int64Index([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 1
5, 16,
            17, 18, 19, 20, 21, 22, 23, 24, 25],
           dtype='int64')
```

In [14]:
```
headers = ["symboling","normalized-losses","make","fuel-type","aspiration",
         "drive-wheels","engine-location","wheel-base", "length","width","h
         "num-of-cylinders", "engine-size","fuel-system","bore","stroke","c
         "peak-rpm","city-mpg","highway-mpg","price"]
```

In [15]:
```
df.columns=headers
```

In [16]:
```
df.columns
```

Out[16]:
```
Index(['symboling', 'normalized-losses', 'make', 'fuel-type', 'aspiratio
n',
       'num-of-doors', 'body-style', 'drive-wheels', 'engine-location',
       'wheel-base', 'length', 'width', 'height', 'curb-weight', 'engine-t
ype',
       'num-of-cylinders', 'engine-size', 'fuel-system', 'bore', 'stroke',
       'compression-ratio', 'horsepower', 'peak-rpm', 'city-mpg',
       'highway-mpg', 'price'],
      dtype='object')
```

In [17]:
```
df.head()
```

Out[17]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wh b
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	8
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	8
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	9
3	2	164	audi	gas	std	four	sedan	fwd	front	9
4	2	164	audi	gas	std	four	sedan	4wd	front	9

5 rows × 26 columns

In [18]:
```
df.shape
```

Out[18]:
```
(205, 26)
```

In [19]:
```
df.info()
```

```
<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
```

In [20]:
```
df.describe()
```

Out[20]:

	symboling	wheel-base	length	width	height	curb-weight	engine-size
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317
std	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000
25%	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000
50%	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000
75%	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000
max	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000

In [21]:
```
df.describe(include="all")
```

Out[21]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	
count	205.000000	205	205	205	205	205	205	205	205	20
unique	NaN	52	22	2	2	3	5	3	2	
top	NaN	?	toyota	gas	std	four	sedan	fwd	front	
freq	NaN	41	32	185	168	114	96	120	202	
mean	0.834146	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	9
std	1.245307	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
min	-2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	8
25%	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	9
50%	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	9
75%	2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	10
max	3.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	12

11 rows × 26 columns

In [22]:
```
df["symboling"]
```

Out[22]:
```
0      3
1      3
2      1
3      2
4      2
      ..
200   -1
201   -1
202   -1
203   -1
204   -1
Name: symboling, Length: 205, dtype: int64
```

In [23]:
```
df["symboling"].head()
```

Out[23]:
```
0    3
1    3
2    1
3    2
4    2
Name: symboling, dtype: int64
```

In [24]:
```
df["symboling"].shape
```

Out[24]:
```
(205,)
```

In [25]:
```
df["symboling"].value_counts()
```

Out[25]:
```
 0    67
 1    54
 2    32
 3    27
-1    22
-2     3
Name: symboling, dtype: int64
```

In [26]:
```
df['length']
```

Out[26]:
```
0      168.8
1      168.8
2      171.2
3      176.6
4      176.6
       ...
200    188.8
201    188.8
202    188.8
203    188.8
204    188.8
Name: length, Length: 205, dtype: float64
```

In [27]:
```
df['length'].mean()
```

Out[27]:
```
174.04926829268305
```

In [28]:
```
df['length'].max()
```

Out[28]:
```
208.1
```

In [29]:
```
df['length'].min()
```

Out[29]:
```
141.1
```

In [30]:
```
df['length'].std()
```

Out[30]:
```
12.337288526555186
```

In [31]:
```
df["length"].value_counts()
```

Out[31]:
```
157.3    15
188.8    11
171.7     7
186.7     7
166.3     7
         ..
165.6     1
187.5     1
180.3     1
208.1     1
199.2     1
Name: length, Length: 75, dtype: int64
```

In [32]:
```
df['length'][0]
```

Out[32]:
```
168.8
```

In [33]:
```
df['length'][0:5]
```

Out[33]:
```
0    168.8
1    168.8
2    171.2
3    176.6
4    176.6
Name: length, dtype: float64
```

In [34]:
```
df['length'].head()
```

Out[34]:
```
0    168.8
1    168.8
2    171.2
3    176.6
4    176.6
Name: length, dtype: float64
```

In [35]:
```
import numpy as np
```

In [36]:
```
df.replace("?",np.nan,inplace=True)
```

In [37]:
```
df
```

Out[37]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front
1	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front
2	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front
3	2	164	audi	gas	std	four	sedan	fwd	front
4	2	164	audi	gas	std	four	sedan	4wd	front
...	...	...	...	...	...	...	...	...	...
200	-1	95	volvo	gas	std	four	sedan	rwd	front
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front
202	-1	95	volvo	gas	std	four	sedan	rwd	front
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front

205 rows × 26 columns

In [38]:
```
df_sample=df.replace(np.nan,0,inplace=False)
```

In [39]:
```
df
```

Out[39]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front
1	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front
2	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front
3	2	164	audi	gas	std	four	sedan	fwd	front
4	2	164	audi	gas	std	four	sedan	4wd	front
...	...	...	...	...	...	...	...	...	...
200	-1	95	volvo	gas	std	four	sedan	rwd	front
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front
202	-1	95	volvo	gas	std	four	sedan	rwd	front
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front

205 rows × 26 columns

In [40]:
```
df_sample
```

Out[40]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location
0	3	0	alfa-romero	gas	std	two	convertible	rwd	front
1	3	0	alfa-romero	gas	std	two	convertible	rwd	front
2	1	0	alfa-romero	gas	std	two	hatchback	rwd	front
3	2	164	audi	gas	std	four	sedan	fwd	front
4	2	164	audi	gas	std	four	sedan	4wd	front
...	...	...	...	...	...	...	...	...	...
200	-1	95	volvo	gas	std	four	sedan	rwd	front
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front
202	-1	95	volvo	gas	std	four	sedan	rwd	front
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front

205 rows × 26 columns

In [41]:
```
df["normalized-losses"].dtype
```

Out[41]:
```
dtype('O')
```

In [42]:
```
df["normalized-losses"][0]
```

Out[42]:
```
nan
```

In [43]:
```
df_sample["normalized-losses"][0]
```

Out[43]:
```
0
```

In [44]:
```
df
```

Out[44]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front
1	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front
2	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front
3	2	164	audi	gas	std	four	sedan	fwd	front
4	2	164	audi	gas	std	four	sedan	4wd	front
...	...	...	...	...	...	...	...	...	...
200	-1	95	volvo	gas	std	four	sedan	rwd	front
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front
202	-1	95	volvo	gas	std	four	sedan	rwd	front
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front

205 rows × 26 columns

In [45]:
```
df.head()
```

Out[45]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wh b
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	8
1	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	8
2	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front	9
3	2	164	audi	gas	std	four	sedan	fwd	front	9
4	2	164	audi	gas	std	four	sedan	4wd	front	9

5 rows × 26 columns

In [46]:
```
df.duplicated()
```

Out[46]:
```
0      False
1      False
2      False
3      False
4      False
       ...
200    False
201    False
202    False
203    False
204    False
Length: 205, dtype: bool
```

In [47]:
```
df.duplicated().sum()
```

Out[47]:
```
0
```

In [48]:
```
df
```

Out[48]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front
1	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front
2	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front
3	2	164	audi	gas	std	four	sedan	fwd	front
4	2	164	audi	gas	std	four	sedan	4wd	front
...	...	...	...	...	...	...	...	...	...
200	-1	95	volvo	gas	std	four	sedan	rwd	front
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front
202	-1	95	volvo	gas	std	four	sedan	rwd	front
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front

205 rows × 26 columns

In [49]:
```
df.isnull()
```

Out[49]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel base
0	False	True	False	False	False	False	False	False	False	False
1	False	True	False	False	False	False	False	False	False	False
2	False	True	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...
200	False	False	False	False	False	False	False	False	False	False
201	False	False	False	False	False	False	False	False	False	False
202	False	False	False	False	False	False	False	False	False	False
203	False	False	False	False	False	False	False	False	False	False
204	False	False	False	False	False	False	False	False	False	False

205 rows × 26 columns

In [50]:
```
df["normalized-losses"].isnull()
```

Out[50]:
```
0       True
1       True
2       True
3      False
4      False
       ...
200    False
201    False
202    False
203    False
204    False
Name: normalized-losses, Length: 205, dtype: bool
```

In [51]:
```
df["normalized-losses"].isnull().value_counts()
```

Out[51]:
```
False    164
True      41
Name: normalized-losses, dtype: int64
```

In [52]:
```
missing_df=df.isnull()
```

In [53]:
```
missing_df
```

Out[53]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel base
0	False	True	False	False	False	False	False	False	False	False
1	False	True	False	False	False	False	False	False	False	False
2	False	True	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...
200	False	False	False	False	False	False	False	False	False	False
201	False	False	False	False	False	False	False	False	False	False
202	False	False	False	False	False	False	False	False	False	False
203	False	False	False	False	False	False	False	False	False	False
204	False	False	False	False	False	False	False	False	False	False

205 rows × 26 columns

In [54]:
```
missing_df.sum()
```

Out[54]:
```
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
```

In [55]:
```
df
```

Out[55]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front
1	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front
2	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front
3	2	164	audi	gas	std	four	sedan	fwd	front
4	2	164	audi	gas	std	four	sedan	4wd	front
...	...	...	...	...	...	...	...	...	...
200	-1	95	volvo	gas	std	four	sedan	rwd	front
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front
202	-1	95	volvo	gas	std	four	sedan	rwd	front
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front

205 rows × 26 columns

In [56]:
```
df.dtypes
```

Out[56]:
```
symboling              int64
normalized-losses     object
make                  object
fuel-type             object
aspiration            object
num-of-doors          object
body-style            object
drive-wheels          object
engine-location       object
wheel-base           float64
length               float64
width                float64
height               float64
curb-weight            int64
engine-type           object
num-of-cylinders      object
engine-size            int64
fuel-system           object
bore                  object
stroke                object
compression-ratio    float64
horsepower            object
peak-rpm              object
city-mpg               int64
highway-mpg            int64
price                 object
dtype: object
```

In [57]:
```
df["normalized-losses"]=df["normalized-losses"].astype(float)
```

In [58]:
```
df["normalized-losses"].dtypes
```

Out[58]:
```
dtype('float64')
```

In [59]:
```
df.dtypes
```

Out[59]:
```
symboling              int64
normalized-losses    float64
make                  object
fuel-type             object
aspiration            object
num-of-doors          object
body-style            object
drive-wheels          object
engine-location       object
wheel-base           float64
length               float64
width                float64
height               float64
curb-weight            int64
engine-type           object
num-of-cylinders      object
engine-size            int64
fuel-system           object
bore                  object
stroke                object
compression-ratio    float64
horsepower            object
peak-rpm              object
city-mpg               int64
highway-mpg            int64
price                 object
dtype: object
```

In [60]:
```
df[["bore"]] = df[["bore"]].astype("float")
df[["normalized-losses"]] = df[["normalized-losses"]].astype("float")
df[["price"]] = df[["price"]].astype("float")
df[["peak-rpm"]] = df[["peak-rpm"]].astype("float")

df[["stroke"]]=df[["stroke"]].astype("float")
df[["horsepower"]]=df[["horsepower"]].astype("float")
```

In [61]:
```
df.dtypes
```

Out[61]:
```
symboling              int64
normalized-losses    float64
make                  object
fuel-type             object
aspiration            object
num-of-doors          object
body-style            object
drive-wheels          object
engine-location       object
wheel-base           float64
length               float64
width                float64
height               float64
curb-weight            int64
engine-type           object
num-of-cylinders      object
engine-size            int64
fuel-system           object
bore                 float64
stroke               float64
compression-ratio    float64
horsepower           float64
peak-rpm             float64
city-mpg               int64
highway-mpg            int64
price                float64
dtype: object
```

In [62]:
```
missing_df.sum()
```

Out[62]:
```
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
```

In [63]:
```
missing_df.sum().sum()
```

Out[63]:
```
59
```

```
normalized-losses    40
num-of-doors          2
bore                  4
stroke                4
horsepower            2
peak-rpm              2
price                 4
```

In [64]:
```
df['normalized-losses']
```

Out[64]:
```
0        NaN
1        NaN
2        NaN
3      164.0
4      164.0
       ...
200     95.0
201     95.0
202     95.0
203     95.0
204     95.0
Name: normalized-losses, Length: 205, dtype: float64
```

In [65]:
```
import numpy as np
```

In [66]:
```
avg1=df['normalized-losses'].mean()
```

In [67]:
```
df['normalized-losses'].replace(np.NaN,avg1,inplace=True)
```

In [68]:
```
df['normalized-losses']
```

Out[68]:
```
0      122.0
1      122.0
2      122.0
3      164.0
4      164.0
       ...
200     95.0
201     95.0
202     95.0
203     95.0
204     95.0
Name: normalized-losses, Length: 205, dtype: float64
```

In [69]:
```
df["normalized-losses"].isnull()
```

Out[69]:
```
0      False
1      False
2      False
3      False
4      False
       ...
200    False
201    False
202    False
203    False
204    False
Name: normalized-losses, Length: 205, dtype: bool
```

In [70]:
```
df["normalized-losses"].isnull().sum()
```

Out[70]:
```
0
```

In [71]:
```
df['num-of-doors']
```

Out[71]:
```
0       two
1       two
2       two
3      four
4      four
       ...
200    four
201    four
202    four
203    four
204    four
Name: num-of-doors, Length: 205, dtype: object
```

In [72]:
```
df['num-of-doors'].isnull().sum()
```

Out[72]:
```
2
```

In [73]:
```
df['num-of-doors'].value_counts()
```

Out[73]:
```
four    114
two      89
Name: num-of-doors, dtype: int64
```

In [74]:
```
df['num-of-doors'].replace(np.nan,'four',inplace=True)
```

In [75]:
```
df['num-of-doors'].isnull().sum()
```

Out[75]:
```
0
```

In [76]:
```
df['num-of-doors'].value_counts()
```

Out[76]:
```
four    116
two      89
Name: num-of-doors, dtype: int64
```

In [77]:
```
df["bore"].isnull().sum()
```

Out[77]:
```
4
```

In [78]:
```
df["bore"]
```

Out[78]:
```
0      3.47
1      3.47
2      2.68
3      3.19
4      3.19
       ...
200    3.78
201    3.78
202    3.58
203    3.01
204    3.78
Name: bore, Length: 205, dtype: float64
```

In [79]:
```
x=np.mean(df["bore"])
```

In [80]:
```
df["bore"].replace(np.nan,x,inplace=True)
```

In [81]:
```
df["bore"].isnull().sum()
```

Out[81]:
```
0
```

In [ ]:
```

```

stroke 4 horsepower 2 peak-rpm 2 price 4

In [82]:
```
df["stroke"].replace(np.nan,np.mean(df["stroke"]),inplace=True)
df["stroke"].isnull().sum()
```

Out[82]:
```
0
```

In [83]:
```
df["horsepower"].replace(np.nan,np.mean(df["horsepower"]),inplace=True)
df["horsepower"].isnull().sum()
```

Out[83]:
```
0
```

In [84]:
```
df.shape
```

Out[84]:
```
(205, 26)
```

In [85]:
```
df["peak-rpm"].replace(np.nan,np.mean(df["peak-rpm"]),inplace=True)
df["peak-rpm"].isnull().sum()
```

Out[85]:
```
0
```

In [86]:
```
df["price"].isnull().sum()
```

Out[86]:
```
4
```

In [87]:
```
df.dropna(subset=["price"],axis=0,inplace=True)
```

In [88]:
```
df.reset_index(drop=True,inplace=True)
```

In [89]:
```
df
```

Out[89]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location
0	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front
1	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front
2	1	122.0	alfa-romero	gas	std	two	hatchback	rwd	front
3	2	164.0	audi	gas	std	four	sedan	fwd	front
4	2	164.0	audi	gas	std	four	sedan	4wd	front
...	...	...	...	...	...	...	...	...	...
196	-1	95.0	volvo	gas	std	four	sedan	rwd	front
197	-1	95.0	volvo	gas	turbo	four	sedan	rwd	front
198	-1	95.0	volvo	gas	std	four	sedan	rwd	front
199	-1	95.0	volvo	diesel	turbo	four	sedan	rwd	front
200	-1	95.0	volvo	gas	turbo	four	sedan	rwd	front

201 rows × 26 columns

In [90]:
```
df.shape
```

Out[90]:
```
(201, 26)
```

This is Clean Data Frame

In [91]:
```
df.isnull().sum()
```

Out[91]:
```
symboling            0
normalized-losses    0
make                 0
fuel-type            0
aspiration           0
num-of-doors         0
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                 0
stroke               0
compression-ratio    0
horsepower           0
peak-rpm             0
city-mpg             0
highway-mpg          0
price                0
dtype: int64
```

In [92]:
```
df.isnull().sum().sum()
```

Out[92]:
```
0
```

In [93]:
```
df.dtypes
```

Out[93]:
```
symboling              int64
normalized-losses    float64
make                  object
fuel-type             object
aspiration            object
num-of-doors          object
body-style            object
drive-wheels          object
engine-location       object
wheel-base           float64
length               float64
width                float64
height               float64
curb-weight            int64
engine-type           object
num-of-cylinders      object
engine-size            int64
fuel-system           object
bore                 float64
stroke               float64
compression-ratio    float64
horsepower           float64
peak-rpm             float64
city-mpg               int64
highway-mpg            int64
price                float64
dtype: object
```

In [94]:
```
df['num-of-cylinders']
```

Out[94]:
```
0      four
1      four
2       six
3      four
4      five
       ...
196    four
197    four
198     six
199     six
200    four
Name: num-of-cylinders, Length: 201, dtype: object
```

In [95]:
```
df['length']
```

Out[95]:
```
0      168.8
1      168.8
2      171.2
3      176.6
4      176.6
       ...
196    188.8
197    188.8
198    188.8
199    188.8
200    188.8
Name: length, Length: 201, dtype: float64
```

Data Normalization

In [96]:
```
df['length']=df['length']/df['length'].max()
df['width']=df['width']/df['width'].max()
```

In [97]:
```
df['width']
```

Out[97]:
```
0      0.890278
1      0.890278
2      0.909722
3      0.919444
4      0.922222
         ...
196    0.956944
197    0.955556
198    0.956944
199    0.956944
200    0.956944
Name: width, Length: 201, dtype: float64
```

In [98]:
```
df['width'].head()
```

Out[98]:
```
0    0.890278
1    0.890278
2    0.909722
3    0.919444
4    0.922222
Name: width, dtype: float64
```

In [99]:
```
df
```

Out[99]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location
0	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front
1	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front
2	1	122.0	alfa-romero	gas	std	two	hatchback	rwd	front
3	2	164.0	audi	gas	std	four	sedan	fwd	front
4	2	164.0	audi	gas	std	four	sedan	4wd	front
...	...	...	...	...	...	...	...	...	...
196	-1	95.0	volvo	gas	std	four	sedan	rwd	front
197	-1	95.0	volvo	gas	turbo	four	sedan	rwd	front
198	-1	95.0	volvo	gas	std	four	sedan	rwd	front
199	-1	95.0	volvo	diesel	turbo	four	sedan	rwd	front
200	-1	95.0	volvo	gas	turbo	four	sedan	rwd	front

201 rows × 26 columns

In [100]:
```
df.head()
```

Out[100]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wh b
0	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front	8
1	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front	8
2	1	122.0	alfa-romero	gas	std	two	hatchback	rwd	front	9
3	2	164.0	audi	gas	std	four	sedan	fwd	front	9
4	2	164.0	audi	gas	std	four	sedan	4wd	front	9

5 rows × 26 columns

In [101]:
```
df["city-mpg"]
```

Out[101]:
```
0      21
1      21
2      19
3      24
4      18
       ..
196    23
197    19
198    18
199    26
200    19
Name: city-mpg, Length: 201, dtype: int64
```

In [102]:
```
df.shape
```

Out[102]:
```
(201, 26)
```

Data Standardization

The formula for unit conversion is:

L/100km = 235 / mpg

In [103]:
```
df['city-L/100km'] = 235/df["city-mpg"]
```

In [104]:
```
df.shape
```

Out[104]:
```
(201, 27)
```

In [105]:
```
# Note 1 Coumnn Is Added in df
```

In [106]:
```
df['city-L/100km']
```

Out[106]:
```
0      11.190476
1      11.190476
2      12.368421
3       9.791667
4      13.055556
         ...
196    10.217391
197    12.368421
198    13.055556
199     9.038462
200    12.368421
Name: city-L/100km, Length: 201, dtype: float64
```

In [107]:
```
df["highway-L/100km"] = 235/df["highway-mpg"]
```

In [108]:
```
df.shape
```

Out[108]:
```
(201, 28)
```

In [109]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201 entries, 0 to 200
Data columns (total 28 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   symboling          201 non-null    int64  
 1   normalized-losses  201 non-null    float64
 2   make               201 non-null    object 
 3   fuel-type          201 non-null    object 
 4   aspiration         201 non-null    object 
 5   num-of-doors       201 non-null    object 
 6   body-style         201 non-null    object 
 7   drive-wheels       201 non-null    object 
 8   engine-location    201 non-null    object 
 9   wheel-base         201 non-null    float64
 10  length             201 non-null    float64
 11  width              201 non-null    float64
 12  height             201 non-null    float64
 13  curb-weight        201 non-null    int64  
 14  engine-type        201 non-null    object 
 15  num-of-cylinders   201 non-null    object 
 16  engine-size        201 non-null    int64  
 17  fuel-system        201 non-null    object 
 18  bore               201 non-null    float64
 19  stroke             201 non-null    float64
 20  compression-ratio  201 non-null    float64
 21  horsepower         201 non-null    float64
 22  peak-rpm           201 non-null    float64
 23  city-mpg           201 non-null    int64  
 24  highway-mpg        201 non-null    int64  
 25  price              201 non-null    float64
 26  city-L/100km       201 non-null    float64
 27  highway-L/100km    201 non-null    float64
dtypes: float64(13), int64(5), object(10)
memory usage: 44.1+ KB
```

Remove Extra repeated Columns

1.city-mpg
2.highway-mpg

In [110]:
```
df.drop(["city-mpg"],axis=1,inplace=True)
```

In [111]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201 entries, 0 to 200
Data columns (total 27 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   symboling          201 non-null    int64  
 1   normalized-losses  201 non-null    float64
 2   make               201 non-null    object 
 3   fuel-type          201 non-null    object 
 4   aspiration         201 non-null    object 
 5   num-of-doors       201 non-null    object 
 6   body-style         201 non-null    object 
 7   drive-wheels       201 non-null    object 
 8   engine-location    201 non-null    object 
 9   wheel-base         201 non-null    float64
 10  length             201 non-null    float64
 11  width              201 non-null    float64
 12  height             201 non-null    float64
 13  curb-weight        201 non-null    int64  
 14  engine-type        201 non-null    object 
 15  num-of-cylinders   201 non-null    object 
 16  engine-size        201 non-null    int64  
 17  fuel-system        201 non-null    object 
 18  bore               201 non-null    float64
 19  stroke             201 non-null    float64
 20  compression-ratio  201 non-null    float64
 21  horsepower         201 non-null    float64
 22  peak-rpm           201 non-null    float64
 23  highway-mpg        201 non-null    int64  
 24  price              201 non-null    float64
 25  city-L/100km       201 non-null    float64
 26  highway-L/100km    201 non-null    float64
dtypes: float64(13), int64(4), object(10)
memory usage: 42.5+ KB
```

In [112]:
```
df.drop(["highway-mpg"],axis=1,inplace=True)
```

In [113]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201 entries, 0 to 200
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   symboling          201 non-null    int64  
 1   normalized-losses  201 non-null    float64
 2   make               201 non-null    object 
 3   fuel-type          201 non-null    object 
 4   aspiration         201 non-null    object 
 5   num-of-doors       201 non-null    object 
 6   body-style         201 non-null    object 
 7   drive-wheels       201 non-null    object 
 8   engine-location    201 non-null    object 
 9   wheel-base         201 non-null    float64
 10  length             201 non-null    float64
 11  width              201 non-null    float64
 12  height             201 non-null    float64
 13  curb-weight        201 non-null    int64  
 14  engine-type        201 non-null    object 
 15  num-of-cylinders   201 non-null    object 
 16  engine-size        201 non-null    int64  
 17  fuel-system        201 non-null    object 
 18  bore               201 non-null    float64
 19  stroke             201 non-null    float64
 20  compression-ratio  201 non-null    float64
 21  horsepower         201 non-null    float64
 22  peak-rpm           201 non-null    float64
 23  price              201 non-null    float64
 24  city-L/100km       201 non-null    float64
 25  highway-L/100km    201 non-null    float64
dtypes: float64(13), int64(3), object(10)
memory usage: 41.0+ KB
```

Renaming Column Names

In [114]:
```
df.rename(columns={"num-of-doors":"doors"},inplace=True)
```

In [115]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201 entries, 0 to 200
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   symboling          201 non-null    int64  
 1   normalized-losses  201 non-null    float64
 2   make               201 non-null    object 
 3   fuel-type          201 non-null    object 
 4   aspiration         201 non-null    object 
 5   doors              201 non-null    object 
 6   body-style         201 non-null    object 
 7   drive-wheels       201 non-null    object 
 8   engine-location    201 non-null    object 
 9   wheel-base         201 non-null    float64
 10  length             201 non-null    float64
 11  width              201 non-null    float64
 12  height             201 non-null    float64
 13  curb-weight        201 non-null    int64  
 14  engine-type        201 non-null    object 
 15  num-of-cylinders   201 non-null    object 
 16  engine-size        201 non-null    int64  
 17  fuel-system        201 non-null    object 
 18  bore               201 non-null    float64
 19  stroke             201 non-null    float64
 20  compression-ratio  201 non-null    float64
 21  horsepower         201 non-null    float64
 22  peak-rpm           201 non-null    float64
 23  price              201 non-null    float64
 24  city-L/100km       201 non-null    float64
 25  highway-L/100km    201 non-null    float64
dtypes: float64(13), int64(3), object(10)
memory usage: 41.0+ KB
```

In [116]:
```
df.to_csv('Car_cleaned-data.csv')
```

Data Visulization

In [117]:
```
df.head()
```

Out[117]:

	symboling	normalized-losses	make	fuel-type	aspiration	doors	body-style	drive-wheels	engine-location	wh b
0	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front	8
1	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front	8
2	1	122.0	alfa-romero	gas	std	two	hatchback	rwd	front	9
3	2	164.0	audi	gas	std	four	sedan	fwd	front	9
4	2	164.0	audi	gas	std	four	sedan	4wd	front	9

5 rows × 26 columns

Binning

In [118]:
```
df["horsepower"]
```

Out[118]:
```
0      111.0
1      111.0
2      154.0
3      102.0
4      115.0
       ...
196    114.0
197    160.0
198    134.0
199    106.0
200    114.0
Name: horsepower, Length: 201, dtype: float64
```

In [119]:
```
df["horsepower"].min()
```

Out[119]:
```
48.0
```

In [120]:
```
df["horsepower"].max()
```

Out[120]:
```
262.0
```

In [121]:
```
df["horsepower"].mean()
```

Out[121]:
```
103.40553390682057
```

In [122]:
```
df["horsepower"].describe()
```

Out[122]:
```
count    201.000000
mean     103.405534
std       37.365700
min       48.000000
25%       70.000000
50%       95.000000
75%      116.000000
max      262.000000
Name: horsepower, dtype: float64
```

In [123]:
```
bins = np.linspace(min(df["horsepower"]), max(df["horsepower"]), 4)
```

In [124]:
```
bins
```

Out[124]:
```
array([ 48.        , 119.33333333, 190.66666667, 262.        ])
```

In [125]:
```
a=np.linspace(2,10,4)
```

In [126]:
```
a
```

Out[126]:
```
array([ 2.        ,  4.66666667,  7.33333333, 10.        ])
```

In [127]:
```
group_names = ['Low', 'Medium', 'High']
```

In [128]:
```
df['horsepower-binned'] = pd.cut(df['horsepower'], bins, labels=group_names
```

In [129]:
```
df[['horsepower','horsepower-binned']].head()
```

Out[129]:

	horsepower	horsepower-binned
0	111.0	Low
1	111.0	Low
2	154.0	Medium
3	102.0	Low
4	115.0	Low

In [130]:
```
df["horsepower-binned"].value_counts()
```

Out[130]:
```
Low       153
Medium     43
High        5
Name: horsepower-binned, dtype: int64
```

In [131]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201 entries, 0 to 200
Data columns (total 27 columns):
 #   Column             Non-Null Count  Dtype   
---  ------             --------------  -----   
 0   symboling          201 non-null    int64   
 1   normalized-losses  201 non-null    float64 
 2   make               201 non-null    object  
 3   fuel-type          201 non-null    object  
 4   aspiration         201 non-null    object  
 5   doors              201 non-null    object  
 6   body-style         201 non-null    object  
 7   drive-wheels       201 non-null    object  
 8   engine-location    201 non-null    object  
 9   wheel-base         201 non-null    float64 
 10  length             201 non-null    float64 
 11  width              201 non-null    float64 
 12  height             201 non-null    float64 
 13  curb-weight        201 non-null    int64   
 14  engine-type        201 non-null    object  
 15  num-of-cylinders   201 non-null    object  
 16  engine-size        201 non-null    int64   
 17  fuel-system        201 non-null    object  
 18  bore               201 non-null    float64 
 19  stroke             201 non-null    float64 
 20  compression-ratio  201 non-null    float64 
 21  horsepower         201 non-null    float64 
 22  peak-rpm           201 non-null    float64 
 23  price              201 non-null    float64 
 24  city-L/100km       201 non-null    float64 
 25  highway-L/100km    201 non-null    float64 
 26  horsepower-binned  201 non-null    category
dtypes: category(1), float64(13), int64(3), object(10)
memory usage: 41.3+ KB
```

In [132]:
```
df["horsepower-binned"][0:5]
```

Out[132]:
```
0       Low
1       Low
2    Medium
3       Low
4       Low
Name: horsepower-binned, dtype: category
Categories (3, object): ['Low' < 'Medium' < 'High']
```

Indicator Variable (or Dummy Variable)

In [133]:
```
df["fuel-type"]
```

Out[133]:
```
0         gas
1         gas
2         gas
3         gas
4         gas
        ...
196       gas
197       gas
198       gas
199    diesel
200       gas
Name: fuel-type, Length: 201, dtype: object
```

In [134]:
```
df["fuel-type"].value_counts()
```

Out[134]:
```
gas       181
diesel     20
Name: fuel-type, dtype: int64
```

In [135]:
```
dummy_variable_1 = pd.get_dummies(df["fuel-type"])
```

In [136]:
```
dummy_variable_1.dtypes
```

Out[136]:
```
diesel    uint8
gas       uint8
dtype: object
```

In [137]:
```
dummy_variable_1.head()
```

Out[137]:

	diesel	gas
0	0	1
1	0	1
2	0	1
3	0	1
4	0	1

In [138]:
```
dummy_variable_1["diesel"]
```

Out[138]:
```
0      0
1      0
2      0
3      0
4      0
      ..
196    0
197    0
198    0
199    1
200    0
Name: diesel, Length: 201, dtype: uint8
```

In [139]:
```
dummy_variable_1.rename(columns={'gas':'fuel-type-gas', 'diesel':'fuel-type
```

In [140]:
```
dummy_variable_1.head()
```

Out[140]:

	fuel-type-diesel	fuel-type-gas
0	0	1
1	0	1
2	0	1
3	0	1
4	0	1

merge data frame "df" and "dummy_variable_1"

In [141]:
```
df = pd.concat([df, dummy_variable_1], axis=1)
```

In [142]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201 entries, 0 to 200
Data columns (total 29 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   symboling          201 non-null    int64
 1   normalized-losses  201 non-null    float64
 2   make               201 non-null    object
 3   fuel-type          201 non-null    object
 4   aspiration         201 non-null    object
 5   doors              201 non-null    object
 6   body-style         201 non-null    object
 7   drive-wheels       201 non-null    object
 8   engine-location    201 non-null    object
 9   wheel-base         201 non-null    float64
 10  length             201 non-null    float64
 11  width              201 non-null    float64
 12  height             201 non-null    float64
 13  curb-weight        201 non-null    int64
 14  engine-type        201 non-null    object
 15  num-of-cylinders   201 non-null    object
 16  engine-size        201 non-null    int64
 17  fuel-system        201 non-null    object
 18  bore               201 non-null    float64
 19  stroke             201 non-null    float64
 20  compression-ratio  201 non-null    float64
 21  horsepower         201 non-null    float64
 22  peak-rpm           201 non-null    float64
 23  price              201 non-null    float64
 24  city-L/100km       201 non-null    float64
 25  highway-L/100km    201 non-null    float64
 26  horsepower-binned  201 non-null    category
 27  fuel-type-diesel   201 non-null    uint8
 28  fuel-type-gas      201 non-null    uint8
dtypes: category(1), float64(13), int64(3), object(10), uint8(2)
memory usage: 41.7+ KB
```

drop original column "fuel-type" from "df"

In [143]:
```
df.drop("fuel-type", axis = 1, inplace=True)
```

In [144]:
```
df.head()
```

Out[144]:

	symboling	normalized-losses	make	aspiration	doors	body-style	drive-wheels	engine-location	wheel-base	
0	3	122.0	alfa-romero	std	two	convertible	rwd	front	88.6	(
1	3	122.0	alfa-romero	std	two	convertible	rwd	front	88.6	(
2	1	122.0	alfa-romero	std	two	hatchback	rwd	front	94.5	0
3	2	164.0	audi	std	four	sedan	fwd	front	99.8	0
4	2	164.0	audi	std	four	sedan	4wd	front	99.4	0

5 rows × 28 columns

In [145]:
```
df.info()
```

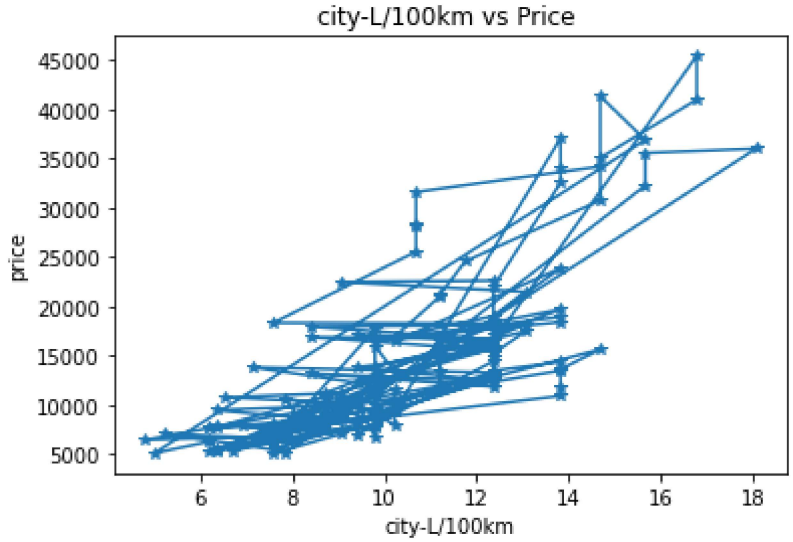
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201 entries, 0 to 200
Data columns (total 28 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   symboling          201 non-null    int64
 1   normalized-losses  201 non-null    float64
 2   make               201 non-null    object
 3   aspiration         201 non-null    object
 4   doors              201 non-null    object
 5   body-style         201 non-null    object
 6   drive-wheels       201 non-null    object
 7   engine-location    201 non-null    object
 8   wheel-base         201 non-null    float64
 9   length             201 non-null    float64
 10  width              201 non-null    float64
 11  height             201 non-null    float64
 12  curb-weight        201 non-null    int64
 13  engine-type        201 non-null    object
 14  num-of-cylinders   201 non-null    object
 15  engine-size        201 non-null    int64
 16  fuel-system        201 non-null    object
 17  bore               201 non-null    float64
 18  stroke             201 non-null    float64
 19  compression-ratio  201 non-null    float64
 20  horsepower         201 non-null    float64
 21  peak-rpm           201 non-null    float64
 22  price              201 non-null    float64
 23  city-L/100km       201 non-null    float64
 24  highway-L/100km    201 non-null    float64
 25  horsepower-binned  201 non-null    category
 26  fuel-type-diesel   201 non-null    uint8
 27  fuel-type-gas      201 non-null    uint8
dtypes: category(1), float64(13), int64(3), object(9), uint8(2)
memory usage: 40.1+ KB
```

Line Chart

In [146]:
```
import matplotlib.pyplot as plt
```

In [147]:
```
import matplotlib.pyplot as plt
x=df["city-L/100km"]
y=df["price"]
plt.plot(x,y,marker='*')
plt.xlabel('city-L/100km')
plt.ylabel('price')

plt.title("city-L/100km vs Price ")
```

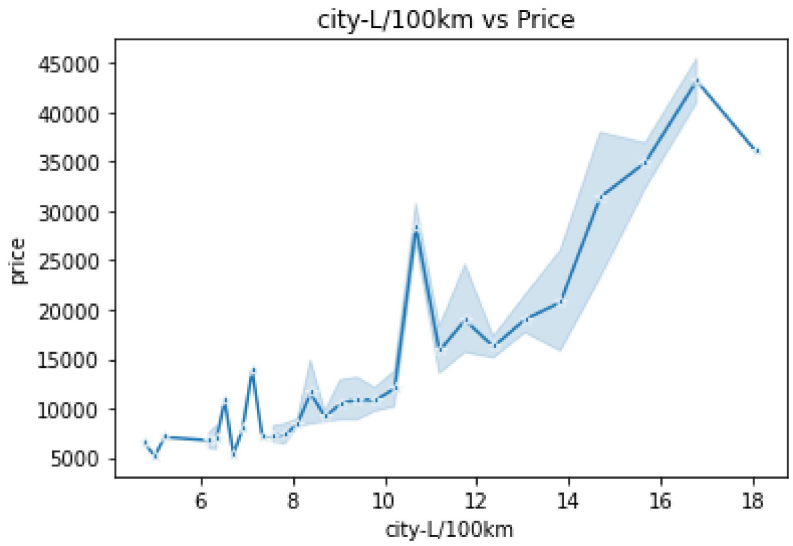
Out[147]: Text(0.5, 1.0, 'city-L/100km vs Price ')



In [148]:
```
import seaborn as sns
x=df["city-L/100km"]
y=df["price"]

sns.lineplot(x,y,marker='*')

plt.xlabel('city-L/100km')
plt.ylabel('price')

plt.title("city-L/100km vs Price ")
```

```
C:\Users\20356119\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: F
utureWarning: Pass the following variables as keyword args: x, y. From ver
sion 0.12, the only valid positional argument will be `data`, and passing
other arguments without an explicit keyword will result in an error or mis
interpretation.
  warnings.warn(
```

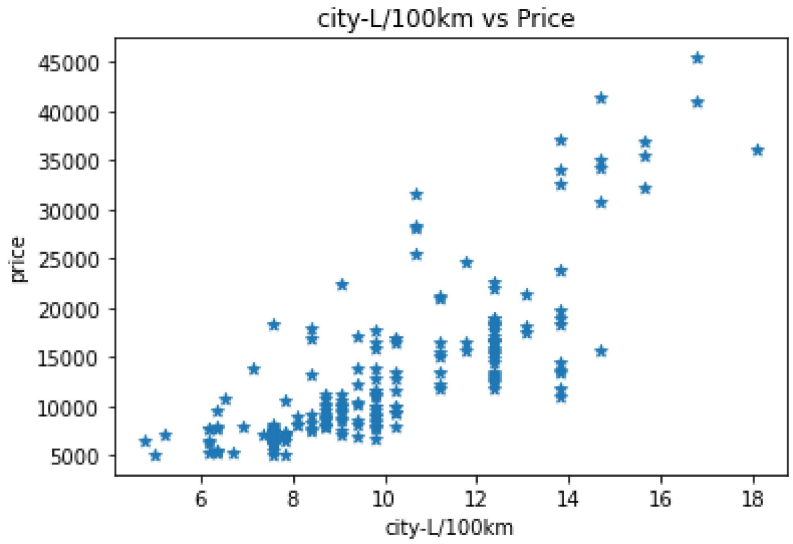
Out[148]:
```
Text(0.5, 1.0, 'city-L/100km vs Price ')
```



Scatter Plot Using Matplotlib

In [149]:
```
import matplotlib.pyplot as plt
x=df["city-L/100km"]
y=df["price"]

plt.scatter(x,y,marker='*',)

plt.xlabel('city-L/100km')

plt.ylabel('price')

plt.title("city-L/100km vs Price ")
```

Out[149]:
```
Text(0.5, 1.0, 'city-L/100km vs Price ')
```



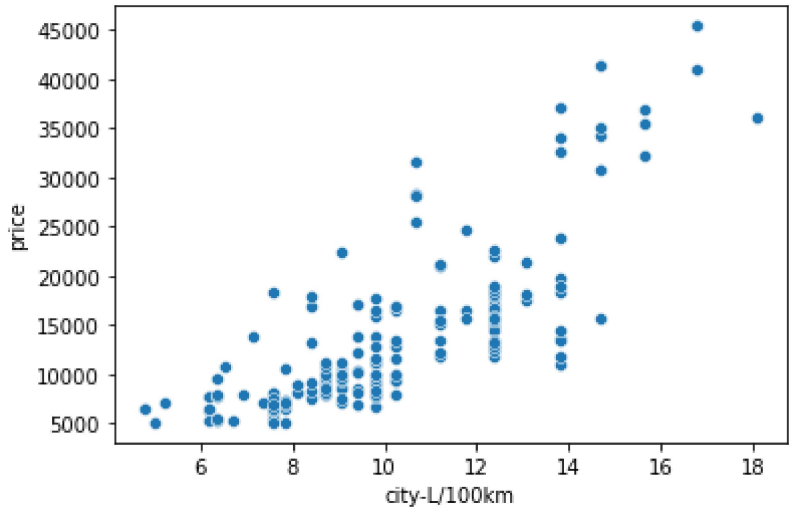
Scatter Plot Using Seaborn

In [150]:
```
import seaborn as sns
```

In [151]:
```
x=df["city-L/100km"]
y=df["price"]
sns.scatterplot(x,y)
```

```
C:\Users\20356119\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: F
utureWarning: Pass the following variables as keyword args: x, y. From ver
sion 0.12, the only valid positional argument will be `data`, and passing
other arguments without an explicit keyword will result in an error or mis
interpretation.
  warnings.warn(
```

Out[151]: <AxesSubplot:xlabel='city-L/100km', ylabel='price'>

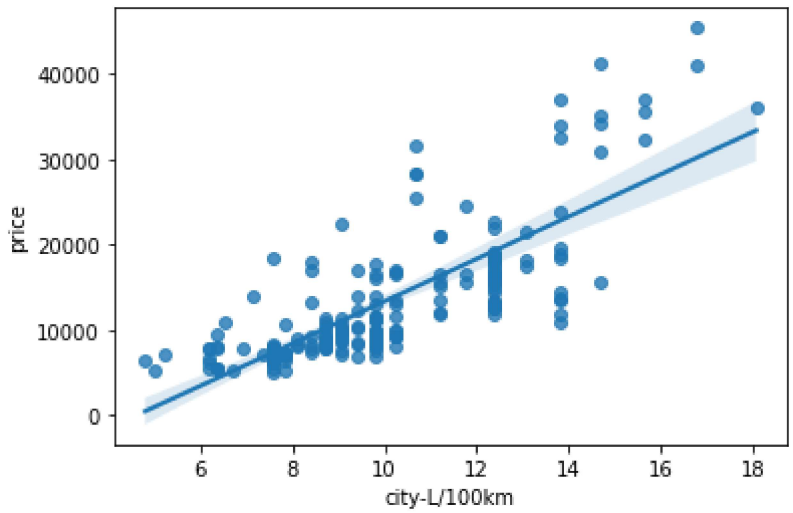


In [152]:
```
import seaborn as sns
```

Regression Plot

In [153]:
```
sns.regplot(x="city-L/100km", y="price", data=df)
```

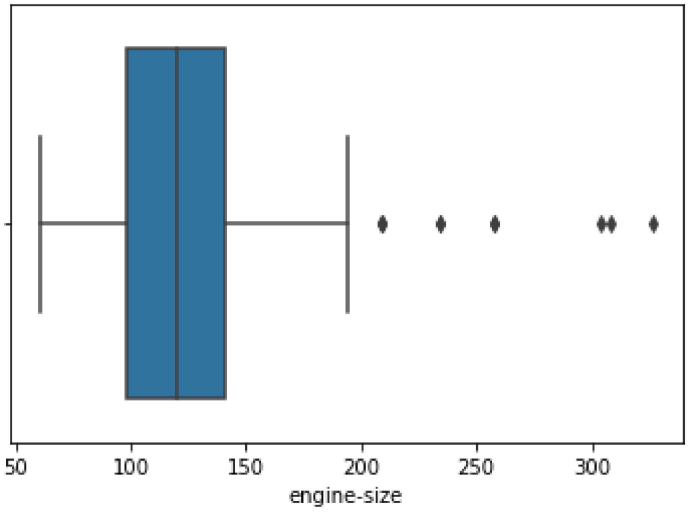
Out[153]: <AxesSubplot:xlabel='city-L/100km', ylabel='price'>



Box Plot

In [154]:
```
#Plot Boxplot of EngineSize
sns.boxplot(x = 'engine-size', data = df)
```

Out[154]:
```
<AxesSubplot:xlabel='engine-size'>
```



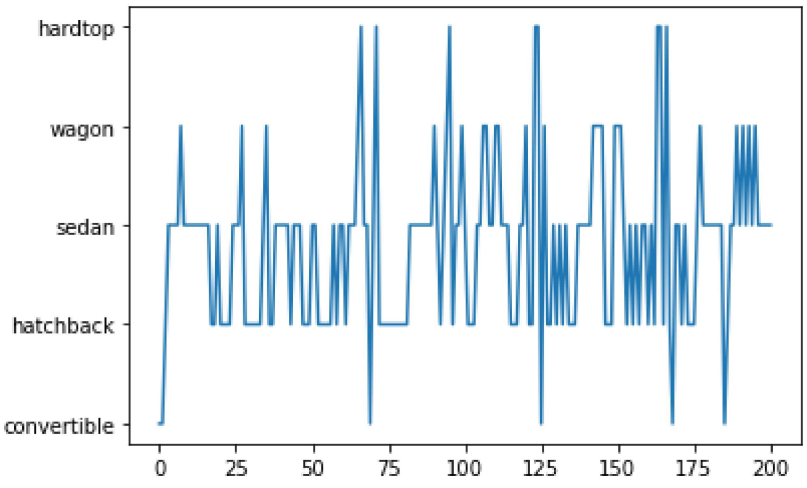
Histplot

In [155]:
```
df["body-style"].value_counts()
```

Out[155]:
```
sedan          94
hatchback      68
wagon          25
hardtop         8
convertible     6
Name: body-style, dtype: int64
```

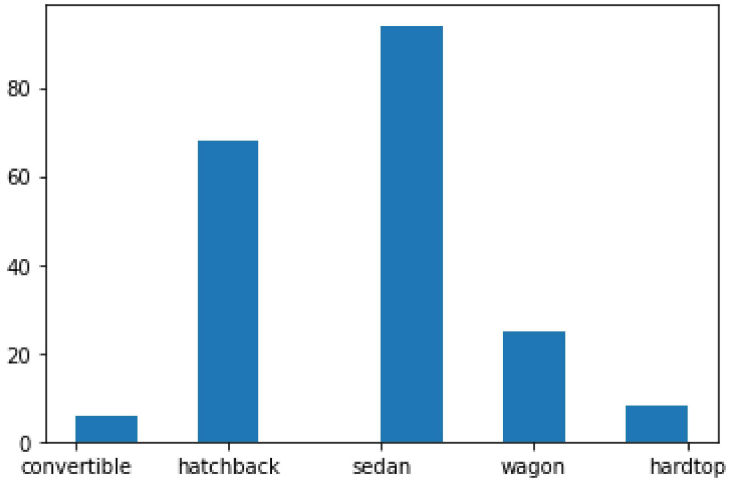
In [156]:
```
plt.plot(df["body-style"])
```

Out[156]:
```
[<matplotlib.lines.Line2D at 0x1f5515390d0>]
```



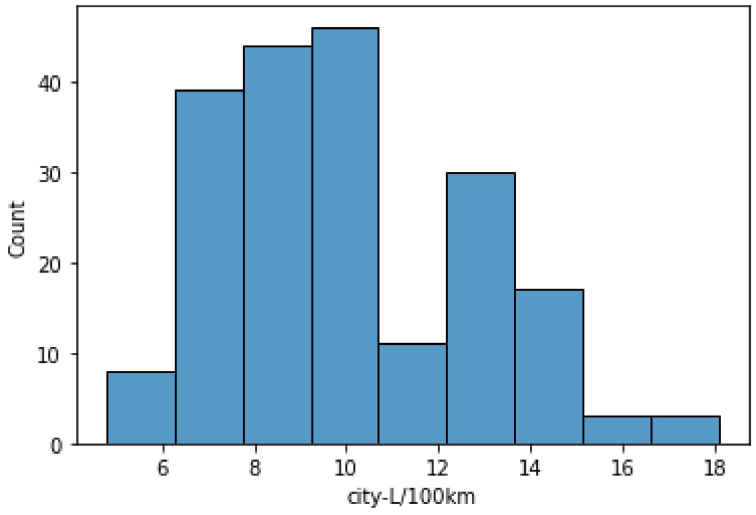
In [157]:
```
plt.hist(df["body-style"])
```

Out[157]:
```
(array([ 6.,  0., 68.,  0.,  0., 94.,  0., 25.,  0.,  8.]),
 array([0. , 0.4, 0.8, 1.2, 1.6, 2. , 2.4, 2.8, 3.2, 3.6, 4. ]),
 <BarContainer object of 10 artists>)
```



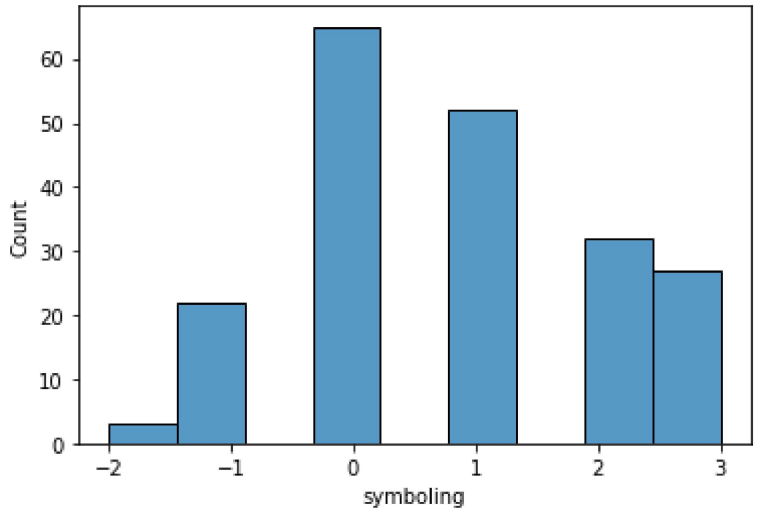
In [158]:
```
sns.histplot(x="city-L/100km",data=df)
```

Out[158]:
```
<AxesSubplot:xlabel='city-L/100km', ylabel='Count'>
```



In [159]:
```
sns.histplot(df["symboling"])
```

Out[159]:
```
<AxesSubplot:xlabel='symboling', ylabel='Count'>
```

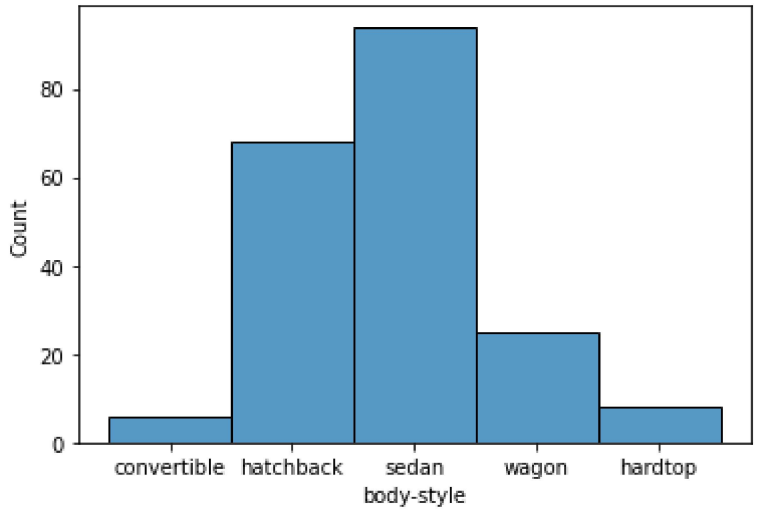


In [160]:
```
df["body-style"].value_counts()
```

Out[160]:
```
sedan          94
hatchback      68
wagon          25
hardtop         8
convertible     6
Name: body-style, dtype: int64
```

In [161]:
```
sns.histplot(df["body-style"])
```

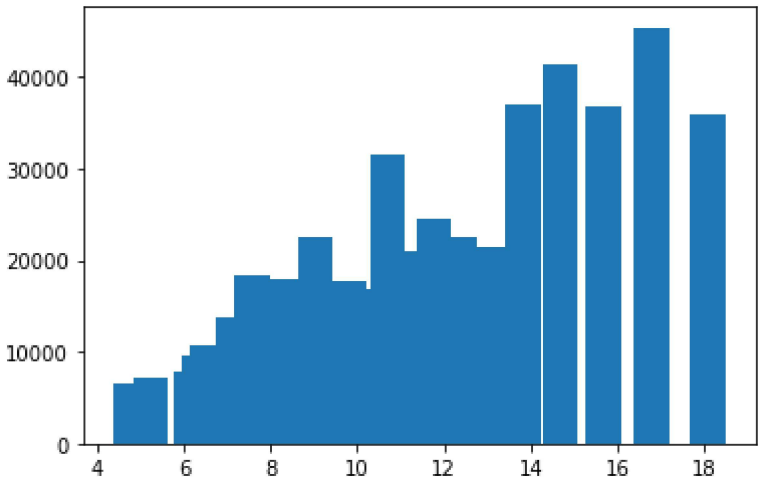
Out[161]:
```
<AxesSubplot:xlabel='body-style', ylabel='Count'>
```



Bar Plot

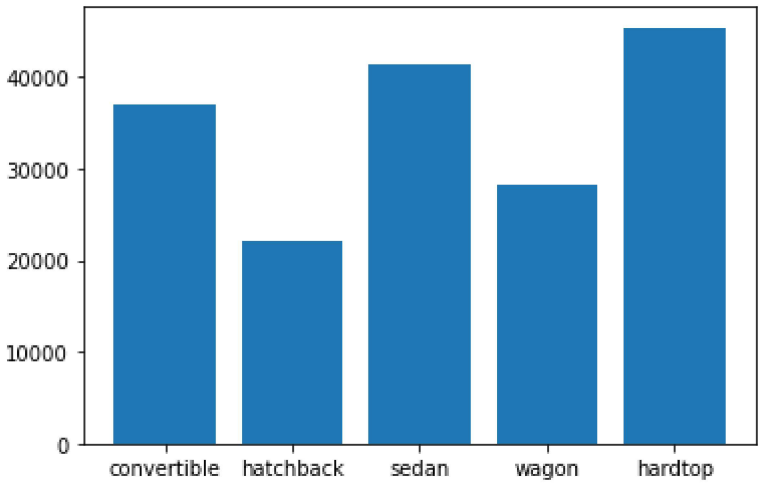
In [162]:
```
x=df["city-L/100km"]
y=df["price"]
plt.bar(x,y)
```

Out[162]: <BarContainer object of 201 artists>



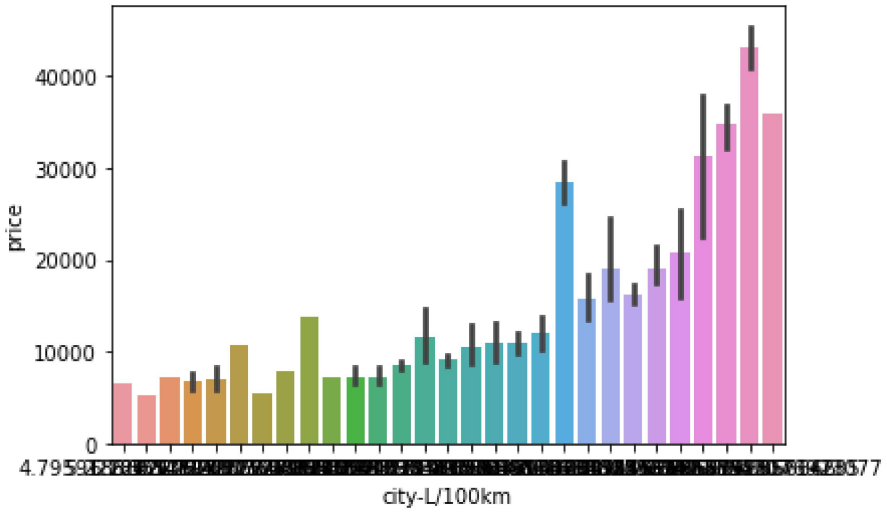
In [163]:
```
x=df["body-style"]
y=df["price"]
plt.bar(x,y)
```

Out[163]: <BarContainer object of 201 artists>



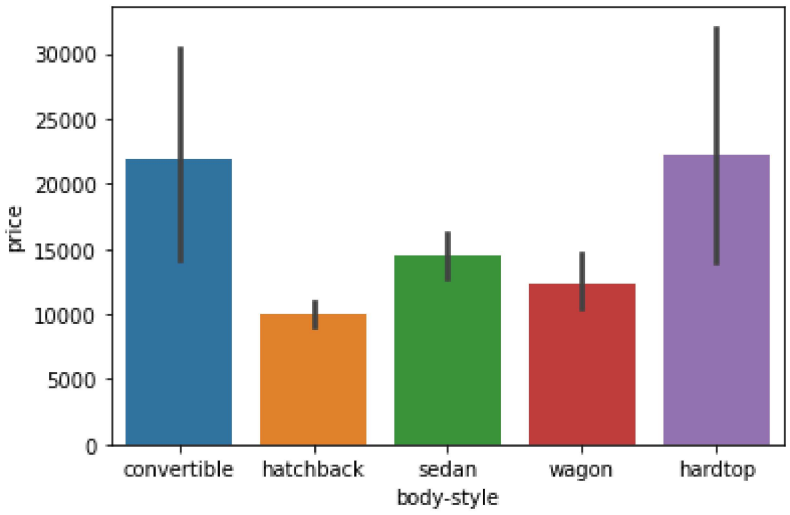
In [164]:
```
sns.barplot(x=df["city-L/100km"], y=df["price"])
```

Out[164]: <AxesSubplot:xlabel='city-L/100km', ylabel='price'>



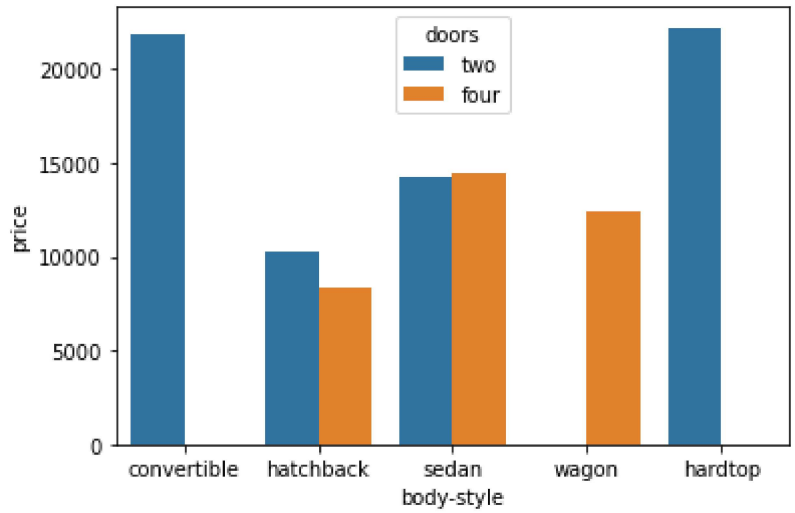
In [165]:
```
sns.barplot(x="body-style", y="price",data=df)
```

Out[165]: <AxesSubplot:xlabel='body-style', ylabel='price'>



In [166]:
```
sns.barplot(x="body-style", y="price", hue="doors",data=df,ci=False)
```

Out[166]:
```
<AxesSubplot:xlabel='body-style', ylabel='price'>
```



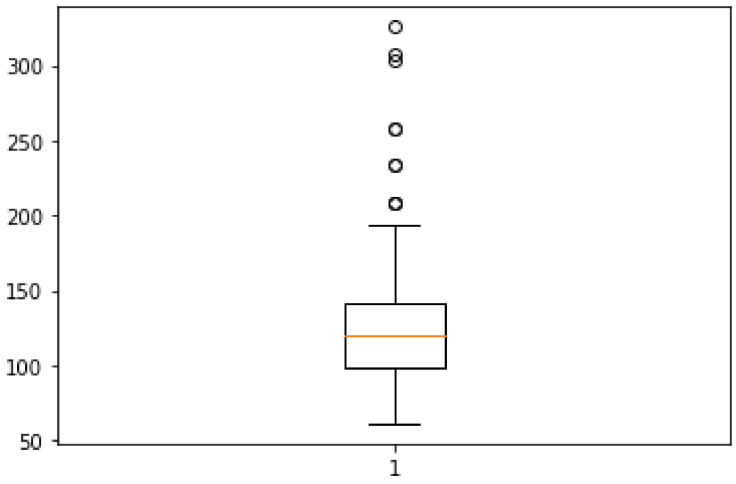
Box Plot

In [167]:
```
df["engine-size"].describe()
```

Out[167]:
```
count    201.000000
mean     126.875622
std       41.546834
min       61.000000
25%       98.000000
50%      120.000000
75%      141.000000
max      326.000000
Name: engine-size, dtype: float64
```

In [168]:
```
plt.boxplot(df["engine-size"])
```

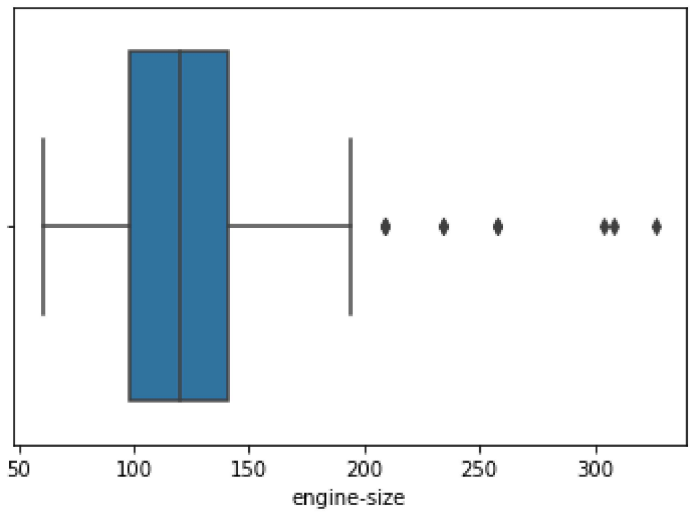
Out[168]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1f552d48a60>,
  <matplotlib.lines.Line2D at 0x1f552d48df0>],
 'caps': [<matplotlib.lines.Line2D at 0x1f552d551c0>,
  <matplotlib.lines.Line2D at 0x1f552d55550>],
 'boxes': [<matplotlib.lines.Line2D at 0x1f552d48730>],
 'medians': [<matplotlib.lines.Line2D at 0x1f552d55940>],
 'fliers': [<matplotlib.lines.Line2D at 0x1f552d55cd0>],
 'means': []}
```



In [169]:
```
sns.boxplot(df["engine-size"])
```

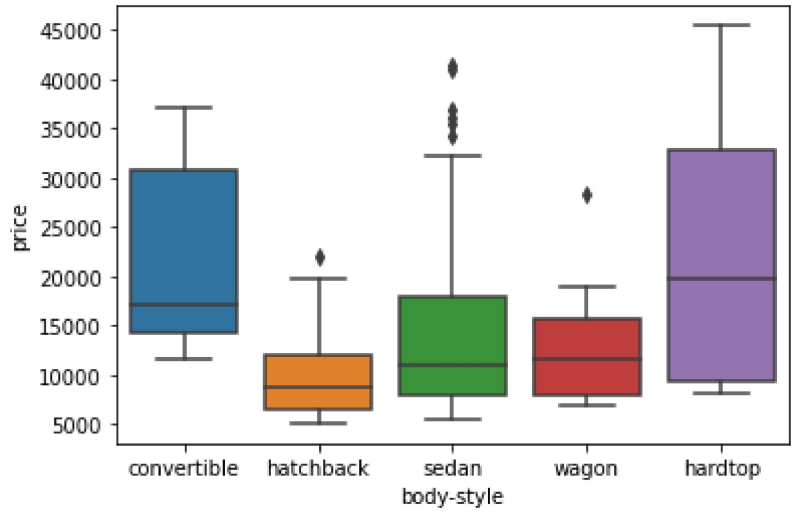
```
C:\Users\20356119\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: F
utureWarning: Pass the following variable as a keyword arg: x. From versio
n 0.12, the only valid positional argument will be `data`, and passing oth
er arguments without an explicit keyword will result in an error or misint
erpretation.
  warnings.warn(
```

Out[169]:
```
<AxesSubplot:xlabel='engine-size'>
```



In [170]:
```
sns.boxplot(x=df["body-style"],y=df["price"])
```

Out[170]:
```
<AxesSubplot:xlabel='body-style', ylabel='price'>
```



In [171]:
```
df.head()
```

Out[171]:

	symboling	normalized-losses	make	aspiration	doors	body-style	drive-wheels	engine-location	wheel-base	
0	3	122.0	alfa-romero	std	two	convertible	rwd	front	88.6	(
1	3	122.0	alfa-romero	std	two	convertible	rwd	front	88.6	(
2	1	122.0	alfa-romero	std	two	hatchback	rwd	front	94.5	0
3	2	164.0	audi	std	four	sedan	fwd	front	99.8	0
4	2	164.0	audi	std	four	sedan	4wd	front	99.4	0

5 rows × 28 columns

In [172]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201 entries, 0 to 200
Data columns (total 28 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   symboling          201 non-null    int64
 1   normalized-losses  201 non-null    float64
 2   make               201 non-null    object
 3   aspiration         201 non-null    object
 4   doors              201 non-null    object
 5   body-style         201 non-null    object
 6   drive-wheels       201 non-null    object
 7   engine-location    201 non-null    object
 8   wheel-base         201 non-null    float64
 9   length             201 non-null    float64
 10  width              201 non-null    float64
 11  height             201 non-null    float64
 12  curb-weight        201 non-null    int64
 13  engine-type        201 non-null    object
 14  num-of-cylinders   201 non-null    object
 15  engine-size        201 non-null    int64
 16  fuel-system        201 non-null    object
 17  bore               201 non-null    float64
 18  stroke             201 non-null    float64
 19  compression-ratio  201 non-null    float64
 20  horsepower         201 non-null    float64
 21  peak-rpm           201 non-null    float64
 22  price              201 non-null    float64
 23  city-L/100km       201 non-null    float64
 24  highway-L/100km    201 non-null    float64
 25  horsepower-binned  201 non-null    category
 26  fuel-type-diesel   201 non-null    uint8
 27  fuel-type-gas      201 non-null    uint8
dtypes: category(1), float64(13), int64(3), object(9), uint8(2)
memory usage: 40.1+ KB
```

Saving Clened Dataset for Further Analysis

In [173]:
```
df.to_csv('cleaned_car_data.csv')
```

Machine Learning

Use cleaned_car_data.csv

In [174]:
```
df=pd.read_csv('cleaned_car_data.csv')
```

In [175]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201 entries, 0 to 200
Data columns (total 29 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Unnamed: 0         201 non-null    int64  
 1   symboling          201 non-null    int64  
 2   normalized-losses  201 non-null    float64
 3   make               201 non-null    object 
 4   aspiration         201 non-null    object 
 5   doors              201 non-null    object 
 6   body-style         201 non-null    object 
 7   drive-wheels       201 non-null    object 
 8   engine-location    201 non-null    object 
 9   wheel-base         201 non-null    float64
 10  length             201 non-null    float64
 11  width              201 non-null    float64
 12  height             201 non-null    float64
 13  curb-weight        201 non-null    int64  
 14  engine-type        201 non-null    object 
 15  num-of-cylinders   201 non-null    object 
 16  engine-size        201 non-null    int64  
 17  fuel-system        201 non-null    object 
 18  bore               201 non-null    float64
 19  stroke             201 non-null    float64
 20  compression-ratio  201 non-null    float64
 21  horsepower         201 non-null    float64
 22  peak-rpm           201 non-null    float64
 23  price              201 non-null    float64
 24  city-L/100km       201 non-null    float64
 25  highway-L/100km    201 non-null    float64
 26  horsepower-binned  201 non-null    object 
 27  fuel-type-diesel   201 non-null    int64  
 28  fuel-type-gas      201 non-null    int64  
dtypes: float64(13), int64(6), object(10)
memory usage: 45.7+ KB
```

In [176]:
```
df.head()
```

Out[176]:

	Unnamed: 0	symboling	normalized-losses	make	aspiration	doors	body-style	drive-wheels	engine location
0	0	3	122.0	alfa-romero	std	two	convertible	rwd	fron
1	1	3	122.0	alfa-romero	std	two	convertible	rwd	fron
2	2	1	122.0	alfa-romero	std	two	hatchback	rwd	fron
3	3	2	164.0	audi	std	four	sedan	fwd	fron
4	4	2	164.0	audi	std	four	sedan	4wd	fron

5 rows × 29 columns

In [177]:
```
df.drop("Unnamed: 0",axis=1,inplace=True)
```

In [178]:
```
df
```

Out[178]:

	symboling	normalized-losses	make	aspiration	doors	body-style	drive-wheels	engine-location	wheel-base
0	3	122.0	alfa-romero	std	two	convertible	rwd	front	88.6
1	3	122.0	alfa-romero	std	two	convertible	rwd	front	88.6
2	1	122.0	alfa-romero	std	two	hatchback	rwd	front	94.5
3	2	164.0	audi	std	four	sedan	fwd	front	99.8
4	2	164.0	audi	std	four	sedan	4wd	front	99.4
...	...	...	...	...	...	...	...	...	...
196	-1	95.0	volvo	std	four	sedan	rwd	front	109.1
197	-1	95.0	volvo	turbo	four	sedan	rwd	front	109.1
198	-1	95.0	volvo	std	four	sedan	rwd	front	109.1
199	-1	95.0	volvo	turbo	four	sedan	rwd	front	109.1
200	-1	95.0	volvo	turbo	four	sedan	rwd	front	109.1

201 rows × 28 columns

In [179]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201 entries, 0 to 200
Data columns (total 28 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   symboling          201 non-null    int64  
 1   normalized-losses  201 non-null    float64
 2   make               201 non-null    object 
 3   aspiration         201 non-null    object 
 4   doors              201 non-null    object 
 5   body-style         201 non-null    object 
 6   drive-wheels       201 non-null    object 
 7   engine-location    201 non-null    object 
 8   wheel-base         201 non-null    float64
 9   length             201 non-null    float64
 10  width              201 non-null    float64
 11  height             201 non-null    float64
 12  curb-weight        201 non-null    int64  
 13  engine-type        201 non-null    object 
 14  num-of-cylinders   201 non-null    object 
 15  engine-size        201 non-null    int64  
 16  fuel-system        201 non-null    object 
 17  bore               201 non-null    float64
 18  stroke             201 non-null    float64
 19  compression-ratio  201 non-null    float64
 20  horsepower         201 non-null    float64
 21  peak-rpm           201 non-null    float64
 22  price              201 non-null    float64
 23  city-L/100km       201 non-null    float64
 24  highway-L/100km    201 non-null    float64
 25  horsepower-binned  201 non-null    object 
 26  fuel-type-diesel   201 non-null    int64  
 27  fuel-type-gas      201 non-null    int64  
dtypes: float64(13), int64(5), object(10)
memory usage: 44.1+ KB
```

Missing Values Check

In [180]:
```
df.isnull().sum()
```

Out[180]:
```
symboling            0
normalized-losses    0
make                 0
aspiration           0
doors                0
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                 0
stroke               0
compression-ratio    0
horsepower           0
peak-rpm             0
price                0
city-L/100km         0
highway-L/100km      0
horsepower-binned    0
fuel-type-diesel     0
fuel-type-gas        0
dtype: int64
```

In [181]:
```
df.isnull().sum().sum()
```

Out[181]:
```
0
```

Data Type Check

In [182]:
```
df.dtypes
```

Out[182]:
```
symboling              int64
normalized-losses    float64
make                  object
aspiration            object
doors                 object
body-style            object
drive-wheels          object
engine-location       object
wheel-base           float64
length               float64
width                float64
height               float64
curb-weight            int64
engine-type           object
num-of-cylinders      object
engine-size            int64
fuel-system           object
bore                 float64
stroke               float64
compression-ratio    float64
horsepower           float64
peak-rpm             float64
price                float64
city-L/100km         float64
highway-L/100km      float64
horsepower-binned     object
fuel-type-diesel       int64
fuel-type-gas          int64
dtype: object
```

In [183]:
```
df.describe()
```

Out[183]:

	symboling	normalized-losses	wheel-base	length	width	height	curb-weight
count	201.000000	201.00000	201.000000	201.000000	201.000000	201.000000	201.000000
mean	0.840796	122.00000	98.797015	0.837102	0.915126	53.766667	2555.666667
std	1.254802	31.99625	6.066366	0.059213	0.029187	2.447822	517.296727
min	-2.000000	65.00000	86.600000	0.678039	0.837500	47.800000	1488.000000
25%	0.000000	101.00000	94.500000	0.801538	0.890278	52.000000	2169.000000
50%	1.000000	122.00000	97.000000	0.832292	0.909722	54.100000	2414.000000
75%	2.000000	137.00000	102.400000	0.881788	0.925000	55.500000	2926.000000
max	3.000000	256.00000	120.900000	1.000000	1.000000	59.800000	4066.000000

In [184]:
```
df.describe(include='all')
```

Out[184]:

	symboling	normalized-losses	make	aspiration	doors	body-style	drive-wheels	engine-location	whe ba
count	201.000000	201.00000	201	201	201	201	201	201	201.0000
unique	NaN	NaN	22	2	2	5	3	2	Na
top	NaN	NaN	toyota	std	four	sedan	fwd	front	Na
freq	NaN	NaN	32	165	115	94	118	198	Na
mean	0.840796	122.00000	NaN	NaN	NaN	NaN	NaN	NaN	98.7970
std	1.254802	31.99625	NaN	NaN	NaN	NaN	NaN	NaN	6.0663
min	-2.000000	65.00000	NaN	NaN	NaN	NaN	NaN	NaN	86.6000
25%	0.000000	101.00000	NaN	NaN	NaN	NaN	NaN	NaN	94.5000
50%	1.000000	122.00000	NaN	NaN	NaN	NaN	NaN	NaN	97.0000
75%	2.000000	137.00000	NaN	NaN	NaN	NaN	NaN	NaN	102.4000
max	3.000000	256.00000	NaN	NaN	NaN	NaN	NaN	NaN	120.9000

11 rows × 28 columns

Understand The ML Problem

1. Predict a Car Price based on fearures given

Find Correlation wrt price

In [185]:
```
df.corr()
```

Out[185]:

	symboling	normalized-losses	wheel-base	length	width	height	curb-weight
symboling	1.000000	0.466264	-0.535987	-0.365404	-0.242423	-0.550160	-0.233118
normalized-losses	0.466264	1.000000	-0.056661	0.019424	0.086802	-0.373737	0.099404
wheel-base	-0.535987	-0.056661	1.000000	0.876024	0.814507	0.590742	0.782097
length	-0.365404	0.019424	0.876024	1.000000	0.857170	0.492063	0.880665
width	-0.242423	0.086802	0.814507	0.857170	1.000000	0.306002	0.866201
height	-0.550160	-0.373737	0.590742	0.492063	0.306002	1.000000	0.307581
curb-weight	-0.233118	0.099404	0.782097	0.880665	0.866201	0.307581	1.000000
engine-size	-0.110581	0.112360	0.572027	0.685025	0.729436	0.074694	0.849072
bore	-0.140019	-0.029862	0.493244	0.608971	0.544885	0.180449	0.644060
stroke	-0.008153	0.055045	0.158018	0.123952	0.188822	-0.060663	0.167438
compression-ratio	-0.182196	-0.114713	0.250313	0.159733	0.189867	0.259737	0.156433
horsepower	0.075819	0.217299	0.371147	0.579821	0.615077	-0.087027	0.757976
peak-rpm	0.279740	0.239543	-0.360305	-0.285970	-0.245800	-0.309974	-0.279361
price	-0.082391	0.133999	0.584642	0.690628	0.751265	0.135486	0.834415
city-L/100km	0.066171	0.238567	0.476153	0.657373	0.673363	0.003811	0.785353
highway-L/100km	-0.029807	0.181189	0.577576	0.707108	0.736728	0.084301	0.836921
fuel-type-diesel	-0.196735	-0.101546	0.307237	0.211187	0.244356	0.281578	0.221046
fuel-type-gas	0.196735	0.101546	-0.307237	-0.211187	-0.244356	-0.281578	-0.221046

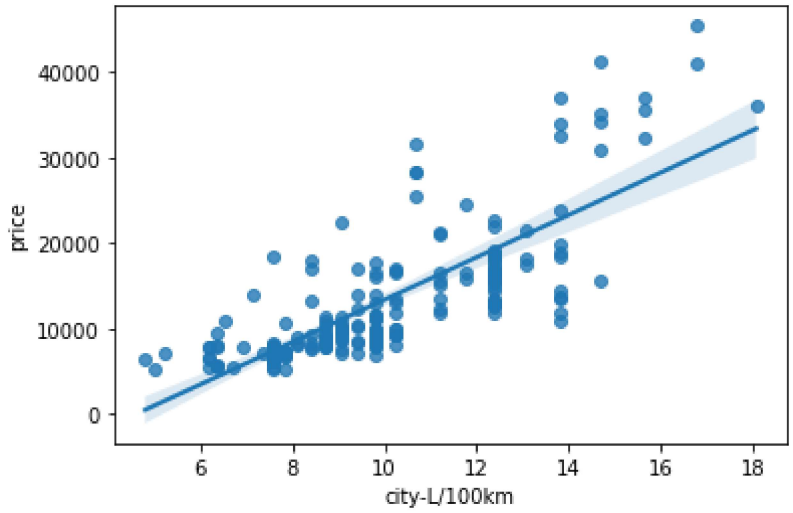
In [186]:
```
df[['city-L/100km', 'price']].corr()
```

Out[186]:

	city-L/100km	price
city-L/100km	1.000000	0.789898
price	0.789898	1.000000

In [187]:
```
sns.regplot(x=df["city-L/100km"], y=df["price"])
```

Out[187]:
```
<AxesSubplot:xlabel='city-L/100km', ylabel='price'>
```



In [188]:
```
df[["highway-L/100km",'price']]
```

Out[188]:

	highway-L/100km	price
0	8.703704	13495.0
1	8.703704	16500.0
2	9.038462	16500.0
3	7.833333	13950.0
4	10.681818	17450.0
...	...	...
196	8.392857	16845.0
197	9.400000	19045.0
198	10.217391	21485.0
199	8.703704	22470.0
200	9.400000	22625.0

201 rows × 2 columns

In [189]:
```
df[["highway-L/100km",'price']].corr()
```

Out[189]:

	highway-L/100km	price
highway-L/100km	1.000000	0.801118
price	0.801118	1.000000

In [190]:
```
df[['length', 'price']].corr()
```

Out[190]:

	length	price
length	1.000000	0.690628
price	0.690628	1.000000

In [191]:
```
df[['width', 'price']].corr()
```

Out[191]:

	width	price
width	1.000000	0.751265
price	0.751265	1.000000

In [192]:
```
df[['curb-weight', 'price']].corr()
```

Out[192]:

	curb-weight	price
curb-weight	1.000000	0.834415
price	0.834415	1.000000

In [193]:
```
df[['engine-size', 'price']].corr()
```

Out[193]:

	engine-size	price
engine-size	1.000000	0.872335
price	0.872335	1.000000

In [194]:
```
df[['bore', 'price']].corr()
```

Out[194]:

	bore	price
bore	1.000000	0.543155
price	0.543155	1.000000

In [195]:
```
df[['horsepower', 'price']].corr()
```

Out[195]:

	horsepower	price
horsepower	1.000000	0.809575
price	0.809575	1.000000

In [196]:
```
df[["fuel-type-diesel","price"]]
```

Out[196]:

	fuel-type-diesel	price
0	0	13495.0
1	0	16500.0
2	0	16500.0
3	0	13950.0
4	0	17450.0
...	...	...
196	0	16845.0
197	0	19045.0
198	0	21485.0
199	1	22470.0
200	0	22625.0

201 rows × 2 columns

In [197]:
```
df["fuel-type-diesel"].value_counts()
```

Out[197]:
```
0    181
1     20
Name: fuel-type-diesel, dtype: int64
```

In [198]:
```
df["fuel-type-diesel"].sum()
```

Out[198]:
```
20
```

There are total 20 Diesel Cars

In [199]:
```
df[["fuel-type-diesel","price"]].corr()
```

Out[199]:

	fuel-type-diesel	price
fuel-type-diesel	1.000000	0.110326
price	0.110326	1.000000

In [200]:
```
df["fuel-type-gas"]
```

Out[200]:
```
0      1
1      1
2      1
3      1
4      1
      ..
196    1
197    1
198    1
199    0
200    1
Name: fuel-type-gas, Length: 201, dtype: int64
```

In [201]:
```
df["fuel-type-gas"].value_counts()
```

Out[201]:
```
1    181
0     20
Name: fuel-type-gas, dtype: int64
```

In [202]:
```
df["fuel-type-gas"].sum()
```

Out[202]:
```
181
```

There are total 120 Gas Cars

In [203]:
```
df[["fuel-type-gas","price"]].corr()
```

Out[203]:

	fuel-type-gas	price
fuel-type-gas	1.000000	-0.110326
price	-0.110326	1.000000

It Looks fuel-type-gas and fuel-type-disel has very low impact on price But Still we would Consider it

Select Features X which are highly Correlated to price

In [204]:
```
X=df[["city-L/100km","highway-L/100km","horsepower","bore","engine-size","c
```

In [205]:
```
X.head()
```

Out[205]:

	city-L/100km	highway-L/100km	horsepower	bore	engine-size	curb-weight	wheel-base	length	width	fue typ ga
0	11.190476	8.703704	111.0	3.47	130	2548	88.6	0.811148	0.890278	
1	11.190476	8.703704	111.0	3.47	130	2548	88.6	0.811148	0.890278	
2	12.368421	9.038462	154.0	2.68	152	2823	94.5	0.822681	0.909722	
3	9.791667	7.833333	102.0	3.19	109	2337	99.8	0.848630	0.919444	
4	13.055556	10.681818	115.0	3.19	136	2824	99.4	0.848630	0.922222	

In [206]:
```
X.head()
```

Out[206]:

	city-L/100km	highway-L/100km	horsepower	bore	engine-size	curb-weight	wheel-base	length	width	fue typ ga
0	11.190476	8.703704	111.0	3.47	130	2548	88.6	0.811148	0.890278	
1	11.190476	8.703704	111.0	3.47	130	2548	88.6	0.811148	0.890278	
2	12.368421	9.038462	154.0	2.68	152	2823	94.5	0.822681	0.909722	
3	9.791667	7.833333	102.0	3.19	109	2337	99.8	0.848630	0.919444	
4	13.055556	10.681818	115.0	3.19	136	2824	99.4	0.848630	0.922222	

In [207]:
```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201 entries, 0 to 200
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   city-L/100km      201 non-null    float64
 1   highway-L/100km   201 non-null    float64
 2   horsepower        201 non-null    float64
 3   bore              201 non-null    float64
 4   engine-size       201 non-null    int64  
 5   curb-weight       201 non-null    int64  
 6   wheel-base        201 non-null    float64
 7   length            201 non-null    float64
 8   width             201 non-null    float64
 9   fuel-type-gas     201 non-null    int64  
 10  fuel-type-diesel  201 non-null    int64  
dtypes: float64(7), int64(4)
memory usage: 17.4 KB
```

In [208]:
```
X.shape
```

Out[208]:
```
(201, 11)
```

Targets

In [209]:
```
Y=df["price"]
```

In [210]:
```
Y.head()
```

Out[210]:
```
0    13495.0
1    16500.0
2    16500.0
3    13950.0
4    17450.0
Name: price, dtype: float64
```

In [211]:
```
Y.shape
```

Out[211]:
```
(201,)
```

Train Test Split

In [212]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.25,random_st
```

In [213]:
```
X_train.shape
```

Out[213]:
```
(150, 11)
```

In [214]:
```
X_test.shape
```

Out[214]:
```
(51, 11)
```

In [215]:
```
Y_train.shape
```

Out[215]:
```
(150,)
```

In [216]:
```
X_train.head()
```

Out[216]:

	city-L/100km	highway-L/100km	horsepower	bore	engine-size	curb-weight	wheel-base	length	width	t
107	9.400000	9.400000	95.0	3.70	152	3430	114.2	0.955790	0.950000	
43	9.791667	8.103448	90.0	3.43	119	2734	96.0	0.829409	0.905556	
153	7.833333	6.351351	70.0	3.19	98	2109	95.7	0.799135	0.894444	
114	13.055556	9.791667	142.0	3.61	134	3130	108.0	0.897165	0.948611	
46	18.076923	13.823529	262.0	3.54	326	3950	102.0	0.921192	0.980556	

In [217]:
```
Y_train.head()
```

Out[217]:
```
107    13860.0
43     11048.0
153     7198.0
114    18150.0
46     36000.0
Name: price, dtype: float64
```

In [218]:
```
Y_train.dtype
```

Out[218]:
```
dtype('float64')
```

Convert features and targets into Array

In [219]:
```
import numpy as np
```

In [220]:
```
X_train_df=np.asanyarray(X_train)
X_test_df=np.asanyarray(X_test)
Y_train_df=np.asanyarray(Y_train)
Y_test_df=np.asanyarray(Y_test)
```

In [221]:
```
X_train_df.shape
```

Out[221]:
```
(150, 11)
```

In [222]:
```
X_test_df.shape
```

Out[222]:
```
(51, 11)
```

In [223]:
```
Y_train_df.shape
```

Out[223]:
```
(150,)
```

In [224]:
```
Y_test_df.shape
```

Out[224]:
```
(51,)
```

In [225]:
```
X_train_df
```

Out[225]:
```
array([[  9.4       ,   9.4       ,  95.        , ...,   0.95      ,
          0.        ,   1.        ],
       [  9.79166667,   8.10344828,  90.        , ...,   0.90555556,
          1.        ,   0.        ],
       [  7.83333333,   6.35135135,  70.        , ...,   0.89444444,
          1.        ,   0.        ],
       ...,
       [  7.58064516,   6.18421053,  68.        , ...,   0.88611111,
          1.        ,   0.        ],
       [  9.4       ,   7.58064516,  88.        , ...,   0.92916667,
          1.        ,   0.        ],
       [ 13.82352941,  10.68181818, 152.        , ...,   0.92361111,
          1.        ,   0.        ]])
```

In [226]:
```
Y_train_df
```

Out[226]:
```
array([13860., 11048.,  7198., 18150., 36000., 15040., 19699., 11245.,
        5572., 11694.,  9495., 10245.,  9980., 21105.,  6189.,  8921.,
       15750.,  6575.,  8495., 28176.,  9960., 18344., 15690., 17199.,
        6849., 41315., 11549., 16925.,  8949., 22625., 16558.,  7895.,
        8249., 17450., 13499.,  7099.,  8189.,  8948., 15985., 14489.,
       17075.,  7738.,  9295., 24565.,  8778.,  8358., 16503.,  6795.,
        5572.,  7295., 40960., 12629.,  7799.,  9549.,  6649.,  6479.,
        7775., 10795., 11850., 35056., 10945.,  7957., 13415., 10698.,
        6669., 11248.,  9095.,  9995.,  7349., 18950., 25552., 11259.,
       32528.,  7995., 18420., 10898., 17669.,  6989.,  9233.,  7689.,
        6529.,  7775., 21485., 18620.,  5195.,  8058., 16430.,  9538.,
       30760.,  7999., 36880.,  7898.,  9279.,  7129.,  8499.,  7299.,
       15510., 34028., 31600., 20970., 16900., 15580., 13499.,  7053.,
       13845.,  6295.,  6338.,  9989.,  8845., 34184., 16500.,  5399.,
       11595., 13200.,  9639., 16695., 18150., 13495., 10595., 15250.,
       19045., 18399., 12170., 10345., 12440.,  7126., 12764.,  5348.,
       37028.,  5499.,  6855., 15998., 28248., 32250.,  8921., 14869.,
        8195., 17950., 10198., 18920.,  9258., 16630.,  7788., 12940.,
       23875.,  5389.,  8013.,  6692., 12290., 14399.])
```

In [227]:
```
Y_test_df
```

Out[227]:
```
array([ 7609.,  9959.,  6918.,  7898., 16515.,  6938., 22018., 17710.,
       12964., 45400.,  9988.,  7295.,  6377., 12945., 15645., 11900.,
       13950., 10295.,  6229.,  7463., 13295.,  8558., 13645., 11199.,
       11845.,  7499.,  8495.,  5151.,  8845.,  9279.,  7609.,  7975.,
        6785.,  7603.,  6695.,  7395., 16845.,  7957., 22470.,  6692.,
        8238., 18280.,  5118., 35550.,  6229., 16500.,  8449.,  6095.,
        9298.,  9895.,  6488.])
```

Create a ML Model using Scikit Learn Library

In [228]:
```
from sklearn import linear_model
model= linear_model.LinearRegression()
```

In [229]:
```
model.fit(X_train_df,Y_train_df)
```

Out[229]:
```
LinearRegression()
```

In [230]:
```
Y_pred=model.predict(X_test)
```

In [231]:
```
model.intercept_
```

Out[231]:
```
-47962.72222914252
```

In [232]:
```
model.coef_
```

Out[232]:
```
array([ 1.77435844e+03, -1.07657824e+03,  2.92376906e+01, -2.97453576e+01,
        8.96880863e+01, -1.01781901e+00,  1.45120497e+02, -1.49252160e+04,
        4.39539017e+04, -2.05451314e+03,  2.05451314e+03])
```

In [233]:
```
Y_pred
```

Out[233]:
```
array([ 5781.81707414, 11921.72250847,  4791.37226423,  5705.50768498,
       13710.31345135,  6942.37265638, 20633.05978213, 19353.76301432,
       18619.04884655, 38625.58038083, 11458.85440454,  6136.83532032,
        6102.13901999,  9764.09863997, 15339.38408517, 15833.50511256,
       11438.41328295,  7210.50823015,  6009.51749004,  8171.72580737,
       16484.11159783,  9639.06241647, 12032.47742402, 14264.55287233,
       12037.56651907,  6019.53164789,  9627.4856333 , -1125.4816478 ,
       11165.84960038, 11931.16729998,  5987.12547181,  9694.66168799,
        7560.63711368,  8498.16187271,  5595.57595266,  5588.99958973,
       16291.68411024,  9703.18501412, 17852.24846341,  5987.12547181,
        6102.66860708, 18818.01952556,  5977.3814992 , 33076.20936898,
        6009.51749004, 12558.96975659, 14406.02971478,  6193.63231676,
        7643.18545455, 14154.30604395,  6029.96948997])
```

In [234]:
```
Y_test
```

Out[234]:
```
119     7609.0
77      9959.0
149     6918.0
150     7898.0
193    16515.0
152     6938.0
122    22018.0
6      17710.0
28     12964.0
71     45400.0
171     9988.0
34      7295.0
21      6377.0
40     12945.0
55     15645.0
104    11900.0
3      13950.0
39     10295.0
117     6229.0
142     7463.0
187    13295.0
26      8558.0
54     13645.0
166    11199.0
53     11845.0
93      7499.0
182     8495.0
17      5151.0
56      8845.0
84      9279.0
25      7609.0
179     7975.0
42      6785.0
136     7603.0
50      6695.0
51      7395.0
196    16845.0
116     7957.0
199    22470.0
24      6692.0
160     8238.0
62     18280.0
134     5118.0
45     35550.0
23      6229.0
1      16500.0
163     8449.0
48      6095.0
161     9298.0
127     9895.0
148     6488.0
Name: price, dtype: float64
```

In [235]:
```
Y_pred[0:5]
```

Out[235]:
```
array([ 5781.81707414, 11921.72250847,  4791.37226423,  5705.50768498,
       13710.31345135])
```

In [236]:
```
Y_test[0:5]
```

Out[236]:
```
119     7609.0
77      9959.0
149     6918.0
150     7898.0
193    16515.0
Name: price, dtype: float64
```

Performance Parameters

Evaluation

We compare the actual values and predicted values to calculate the accuracy of a regression model. Evaluation metrics provide a key role in the development of a model, as it provides insight to areas that require improvement.

There are different model evaluation metrics, lets use MSE here to calculate the accuracy of our model based on the test set:

- Mean Absolute Error: It is the mean of the absolute value of the errors. This is the easiest of the metrics to understand since it's just average error.
- Mean Squared Error (MSE): Mean Squared Error (MSE) is the mean of the squared error. It's more popular than Mean Absolute Error because the focus is geared more towards large errors. This is due to the squared term exponentially increasing larger errors in comparison to smaller ones.
- Root Mean Squared Error (RMSE).
- R-squared is not an error, but rather a popular metric to measure the performance of your regression model. It represents how close the data points are to the fitted regression line. The higher the R-squared value, the better the model fits your data. The best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse).

Mean absolute error

In [237]:
```
mae=np.mean(np.abs(Y_pred-Y_test))
```

In [238]:
```
mae
```

Out[238]:
```
2046.3029357077633
```

Mean Squared error

In [239]:
```
mse=np.mean(np.abs(Y_pred-Y_test)**2)
```

In [240]:
```
mse
```

Out[240]: 6895799.569755058

Root Mean Squared Error

In [241]:
```
rmse=np.sqrt(np.mean((Y_pred-Y_test)**2))
```

In [242]:
```
rmse
```

Out[242]: 2625.9854473616297

In [243]:
```
rmse=np.sqrt(mse)
```

In [244]:
```
rmse
```

Out[244]: 2625.9854473616297

END