Auto Mobile Project

- Uses EDa methos inside Project
- Treat a Missing value
- Treat a Outliers
- Uses a lable encoding

In [1]:
```
## Import the oprating Systen Path
import os
os.getcwd()
```

Out[1]:
```
'C:\\Users\\ap983\\Desktop\\Pandey  IMp\\COMPLETE PYTHON PROJECT\\3.Auto Mobile Project'
```

In [2]:
```
## Impoer All Necessary packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
## Loading the data set
data = pd.read_csv('C:/Users/ap983/Desktop/Pandey  IMp/COMPLETE PYTHON PROJECT/3.Auto Mobile Project//Automobil
data
```

Out[3]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	c
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	3.4	
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	3.15	
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	173	mpfi	3.58	2.87	
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	3.15	

205 rows × 26 columns

In [4]:
```
## Check the top five rows in the data set and ahow the by defauclt five rows
data.head()
```

Out[4]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	comp
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	3.4	
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.4	

5 rows × 26 columns

In [5]:
```
## check Bottom five rows
data.tail()
```

Out[5]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compre
200	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	173	mpfi	3.58	2.87	
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	3.15	

5 rows × 26 columns

In [105]:
```
## check the columns name
data.columns
```

Out[105]:
```
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')
```

In [106]:
```
## Check the  shape for data set
data.shape
```

Out[106]:
```
(205, 26)
```

In [107]:
```
## Check the Information of the date_set  and its dtype
data.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    float64
 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    int32  
 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    float64
 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    float64
 24  highway-mpg        205 non-null    float64
 25  price              205 non-null    object 
dtypes: float64(9), int32(1), int64(2), object(14)
memory usage: 41.0+ KB
```

In [108]:
```
data.isna().sum()
```

Out[108]:
```
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 [109]:
```
## Check the null value uses the isna method or find the boolean value True or False
data.isna().any()
```

Out[109]:
```
symboling            False
normalized-losses    False
make                 False
fuel-type            False
aspiration           False
num-of-doors         False
body-style           False
drive-wheels         False
engine-location      False
wheel-base           False
length               False
width                False
height               False
curb-weight          False
engine-type          False
num-of-cylinders     False
engine-size          False
fuel-system          False
bore                 False
stroke               False
compression-ratio    False
horsepower           False
peak-rpm             False
city-mpg             False
highway-mpg          False
price                False
dtype: bool
```

In [110]:
```
## Check the Any value are empty or not
data.isnull().sum()
```

Out[110]:
```
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 [112]:
```
## Check the satistics for data set
data.describe()
```

Out[112]:

	symboling	normalized-losses	body-style	wheel-base	length	width	height	curb-weight	engine-size	compression-ratio	c
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205
mean	0.834146	119.959024	2.614634	98.724488	174.131805	65.887610	53.724878	2555.565854	124.815610	9.162488	25
std	1.245307	28.376086	0.859081	5.154445	11.193416	1.961127	2.443522	520.680204	32.545885	0.187915	5
min	-2.000000	77.200000	0.000000	93.020000	157.172000	63.600000	47.800000	1488.000000	90.000000	9.000000	16
25%	0.000000	101.000000	2.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	9.000000	19
50%	1.000000	115.000000	3.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	9.000000	24
75%	2.000000	137.000000	3.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	9.400000	30
max	3.000000	182.400000	4.000000	110.000000	195.848000	70.460000	59.800000	4066.000000	201.200000	9.400000	37

In [113]:
```
##print the descreptive table tranpose
data.describe().T
```

Out[113]:

	count	mean	std	min	25%	50%	75%	max
symboling	205.0	0.834146	1.245307	-2.000	0.0	1.0	2.0	3.000
normalized-losses	205.0	119.959024	28.376086	77.200	101.0	115.0	137.0	182.400
body-style	205.0	2.614634	0.859081	0.000	2.0	3.0	3.0	4.000
wheel-base	205.0	98.724488	5.154445	93.020	94.5	97.0	102.4	110.000
length	205.0	174.131805	11.193416	157.172	166.3	173.2	183.1	195.848
width	205.0	65.887610	1.961127	63.600	64.1	65.5	66.9	70.460
height	205.0	53.724878	2.443522	47.800	52.0	54.1	55.5	59.800
curb-weight	205.0	2555.565854	520.680204	1488.000	2145.0	2414.0	2935.0	4066.000
engine-size	205.0	124.815610	32.545885	90.000	97.0	120.0	141.0	201.200
compression-ratio	205.0	9.162488	0.187915	9.000	9.0	9.0	9.4	9.400
city-mpg	205.0	25.087805	5.996903	16.000	19.0	24.0	30.0	37.000
highway-mpg	205.0	30.697756	6.047872	22.000	25.0	30.0	34.0	42.640

In [115]:
```
## replace the question marks with nan value
from numpy import nan
data['normalized-losses'] = data['normalized-losses'].replace('?',np.nan)
```

In [15]:
```
## After replacing check the null value inside the data set
data.isnull().sum()
```

Out[15]:
```
symboling             0
normalized-losses    41
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 [116]:
```
## fill the nan valus using median method
data.fillna(data.median(),inplace=True)
```

```
C:\Users\ap983\AppData\Local\Temp\ipykernel_16032\1292629150.py:2: FutureWarning: Dropping of nuisance column
s in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeE
rror.  Select only valid columns before calling the reduction.
  data.fillna(data.median(),inplace=True)
```

In [18]:
```
data.isnull().sum()
```

Out[18]:
```
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 [21]:
```
data.median()
```

```
C:\Users\ap983\AppData\Local\Temp\ipykernel_16032\4184645713.py:1: FutureWarning: Dropping of nuisance column
s in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeE
rror.  Select only valid columns before calling the reduction.
  data.median()
```

Out[21]:
```
symboling              1.0
normalized-losses    115.0
wheel-base            97.0
length               173.2
width                 65.5
height                54.1
curb-weight         2414.0
engine-size          120.0
compression-ratio      9.0
city-mpg              24.0
highway-mpg           30.0
dtype: float64
```

In [22]:
```
data.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 [28]:
```
## change the value object to float
data['normalized-losses'] =data['normalized-losses'] .astype('float')
```

In [29]:
```
data.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    float64
 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(6), int64(5), object(15)
memory usage: 41.8+ KB
```

In [30]:
```
data.columns
```

Out[30]:
```
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')
```

In [32]:
```
data.head()
```

Out[32]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	com
0	3	115.0	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	
1	3	115.0	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	
2	1	115.0	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	
3	2	164.0	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.4	
4	2	164.0	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.4	

5 rows × 26 columns

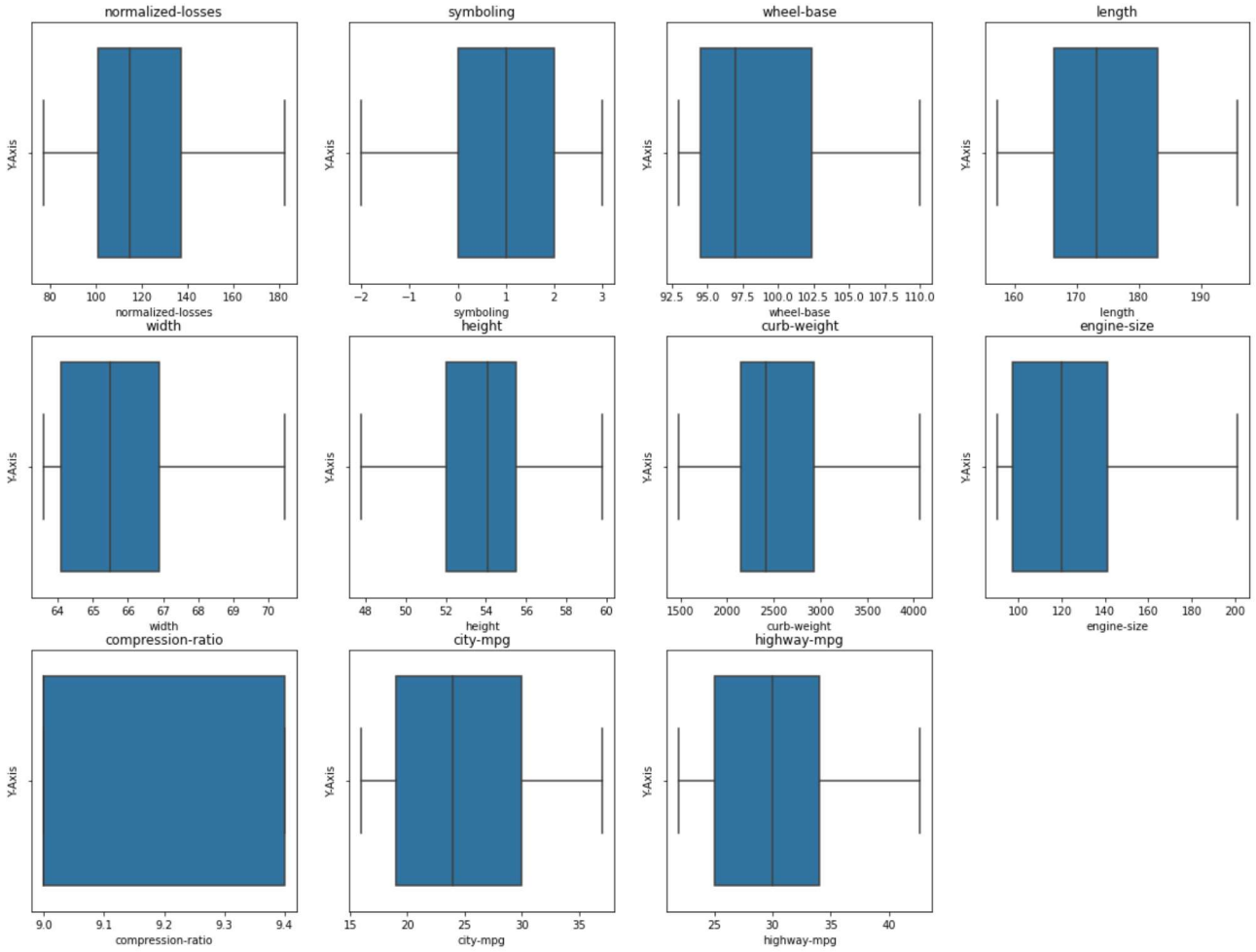
In [37]:
```
data.describe().T
```

Out[37]:

	count	mean	std	min	25%	50%	75%	max
symboling	205.0	0.834146	1.245307	-2.0	0.0	1.0	2.0	3.0
normalized-losses	205.0	120.600000	31.805105	65.0	101.0	115.0	137.0	256.0
wheel-base	205.0	98.756585	6.021776	86.6	94.5	97.0	102.4	120.9
length	205.0	174.049268	12.337289	141.1	166.3	173.2	183.1	208.1
width	205.0	65.907805	2.145204	60.3	64.1	65.5	66.9	72.3
height	205.0	53.724878	2.443522	47.8	52.0	54.1	55.5	59.8
curb-weight	205.0	2555.565854	520.680204	1488.0	2145.0	2414.0	2935.0	4066.0
engine-size	205.0	126.907317	41.642693	61.0	97.0	120.0	141.0	326.0
compression-ratio	205.0	10.142537	3.972040	7.0	8.6	9.0	9.4	23.0
city-mpg	205.0	25.219512	6.542142	13.0	19.0	24.0	30.0	49.0
highway-mpg	205.0	30.751220	6.886443	16.0	25.0	30.0	34.0	54.0

In [68]:
```
## Check the Out liers in differnt columns
plt.figure(figsize=(20,15))
plt.subplot(3,4,1)
plt.title('normalized-losses')
plt.xlabel('X-axis')
plt.ylabel('Y-Axis')
sns.boxplot(data['normalized-losses'])

plt.subplot(3,4,2)
plt.title('symboling')
plt.xlabel('X-axis')
plt.ylabel('Y-Axis')
sns.boxplot(data['symboling'])

plt.subplot(3,4,3)
plt.title('wheel-base')
plt.xlabel('X-axis')
plt.ylabel('Y-Axis')
sns.boxplot(data['wheel-base'])

plt.subplot(3,4,4)
plt.title('length')
plt.xlabel('X-axis')
plt.ylabel('Y-Axis')
sns.boxplot(data['length'])

plt.subplot(3,4,5)
plt.title('width')
plt.xlabel('X-axis')
plt.ylabel('Y-Axis')
sns.boxplot(data['width'])

plt.subplot(3,4,6)
plt.title('height')
plt.xlabel('X-axis')
plt.ylabel('Y-Axis')
sns.boxplot(data['height'])

plt.subplot(3,4,7)
plt.title('curb-weight')
plt.xlabel('X-axis')
plt.ylabel('Y-Axis')
sns.boxplot(data['curb-weight'])

plt.subplot(3,4,8)
plt.title('engine-size')
plt.xlabel('X-axis')
plt.ylabel('Y-Axis')
sns.boxplot(data['engine-size'])

plt.subplot(3,4,9)
plt.title('compression-ratio')
plt.xlabel('X-axis')
plt.ylabel('Y-Axis')
sns.boxplot(data['compression-ratio'])

plt.subplot(3,4,10)
plt.title('city-mpg')
plt.xlabel('X-axis')
plt.ylabel('Y-Axis')
sns.boxplot(data['city-mpg'])

plt.subplot(3,4,11)
plt.title('highway-mpg')
plt.xlabel('X-axis')
plt.ylabel('Y-Axis')
sns.boxplot(data['highway-mpg'])

plt.show()
```

```
C:\Users\ap983\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing ot
her arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\ap983\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing ot
her arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\ap983\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing ot
her arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\ap983\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing ot
her arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\ap983\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing ot
her arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\ap983\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing ot
her arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\ap983\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing ot
her arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\ap983\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing ot
her arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\ap983\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing ot
her arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\ap983\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing ot
her arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\ap983\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing ot
her arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [45]:
```
data.describe().T
```

Out[45]:

	count	mean	std	min	25%	50%	75%	max
symboling	205.0	0.834146	1.245307	-2.0	0.0	1.0	2.0	3.0
normalized-losses	205.0	120.600000	31.805105	65.0	101.0	115.0	137.0	256.0
wheel-base	205.0	98.756585	6.021776	86.6	94.5	97.0	102.4	120.9
length	205.0	174.049268	12.337289	141.1	166.3	173.2	183.1	208.1
width	205.0	65.907805	2.145204	60.3	64.1	65.5	66.9	72.3
height	205.0	53.724878	2.443522	47.8	52.0	54.1	55.5	59.8
curb-weight	205.0	2555.565854	520.680204	1488.0	2145.0	2414.0	2935.0	4066.0
engine-size	205.0	126.907317	41.642693	61.0	97.0	120.0	141.0	326.0
compression-ratio	205.0	10.142537	3.972040	7.0	8.6	9.0	9.4	23.0
city-mpg	205.0	25.219512	6.542142	13.0	19.0	24.0	30.0	49.0
highway-mpg	205.0	30.751220	6.886443	16.0	25.0	30.0	34.0	54.0

In [46]:
```
## Treat the outliers using clip method ,clip method working on lower value to upper values
data['normalized-losses'] = data['normalized-losses'].clip(lower = data['normalized-losses'].quantile(0.05),up
```

In [48]:
```
data['wheel-base'] = data['wheel-base'].clip(lower=data['wheel-base'].quantile(0.05),upper=data['wheel-base'].
```

In [51]:
```
data['length'] = data['length'].clip(lower=data['length'].quantile(0.05),upper=data['length'].quantile(0.95))
```

In [52]:
```
data['width'] = data['width'].clip(lower=data['width'].quantile(0.05),upper=data['width'].quantile(0.95))
```

In [53]:
```
data['engine-size'] = data['engine-size'].clip(lower=data['engine-size'].quantile(0.05),upper=data['engine-size
```

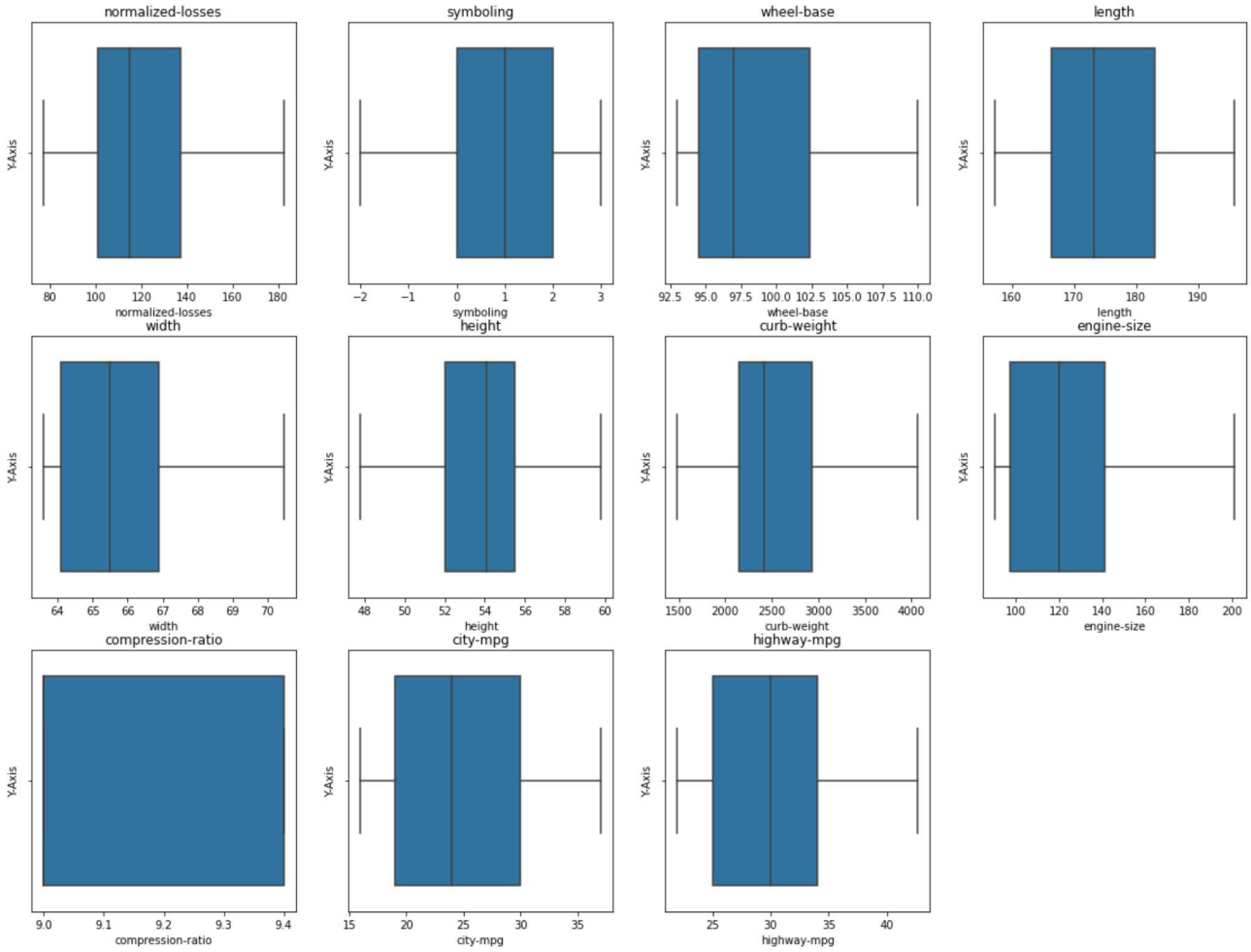
In [54]:
```
] = data['compression-ratio'].clip(lower=data['compression-ratio'].quantile(0.05),upper=data['compression-ratio
```

In [55]:
```
data['city-mpg'] = data['city-mpg'].clip(lower=data['city-mpg'].quantile(0.05),upper=data['city-mpg'].quantile
```

In [56]:
```
data['highway-mpg'] = data['highway-mpg'].clip(lower=data['highway-mpg'].quantile(0.05),upper=data['highway-mpg
```

In [117]:
```
## After teated the outliers print the boxplot
plt.figure(figsize=(20,15))
plt.subplot(3,4,1)
plt.title('normalized-losses')
plt.xlabel('X-axis')
plt.ylabel('Y-Axis')
sns.boxplot(data['normalized-losses'])

plt.subplot(3,4,2)
plt.title('symboling')
plt.xlabel('X-axis')
plt.ylabel('Y-Axis')
sns.boxplot(data['symboling'])

plt.subplot(3,4,3)
plt.title('wheel-base')
plt.xlabel('X-axis')
plt.ylabel('Y-Axis')
sns.boxplot(data['wheel-base'])

plt.subplot(3,4,4)
plt.title('length')
plt.xlabel('X-axis')
plt.ylabel('Y-Axis')
sns.boxplot(data['length'])

plt.subplot(3,4,5)
plt.title('width')
plt.xlabel('X-axis')
plt.ylabel('Y-Axis')
sns.boxplot(data['width'])

plt.subplot(3,4,6)
plt.title('height')
plt.xlabel('X-axis')
plt.ylabel('Y-Axis')
sns.boxplot(data['height'])

plt.subplot(3,4,7)
plt.title('curb-weight')
plt.xlabel('X-axis')
plt.ylabel('Y-Axis')
sns.boxplot(data['curb-weight'])

plt.subplot(3,4,8)
plt.title('engine-size')
plt.xlabel('X-axis')
plt.ylabel('Y-Axis')
sns.boxplot(data['engine-size'])

plt.subplot(3,4,9)
plt.title('compression-ratio')
plt.xlabel('X-axis')
plt.ylabel('Y-Axis')
sns.boxplot(data['compression-ratio'])

plt.subplot(3,4,10)
plt.title('city-mpg')
plt.xlabel('X-axis')
plt.ylabel('Y-Axis')
sns.boxplot(data['city-mpg'])

plt.subplot(3,4,11)
plt.title('highway-mpg')
plt.xlabel('X-axis')
plt.ylabel('Y-Axis')
sns.boxplot(data['highway-mpg'])

plt.show()
```

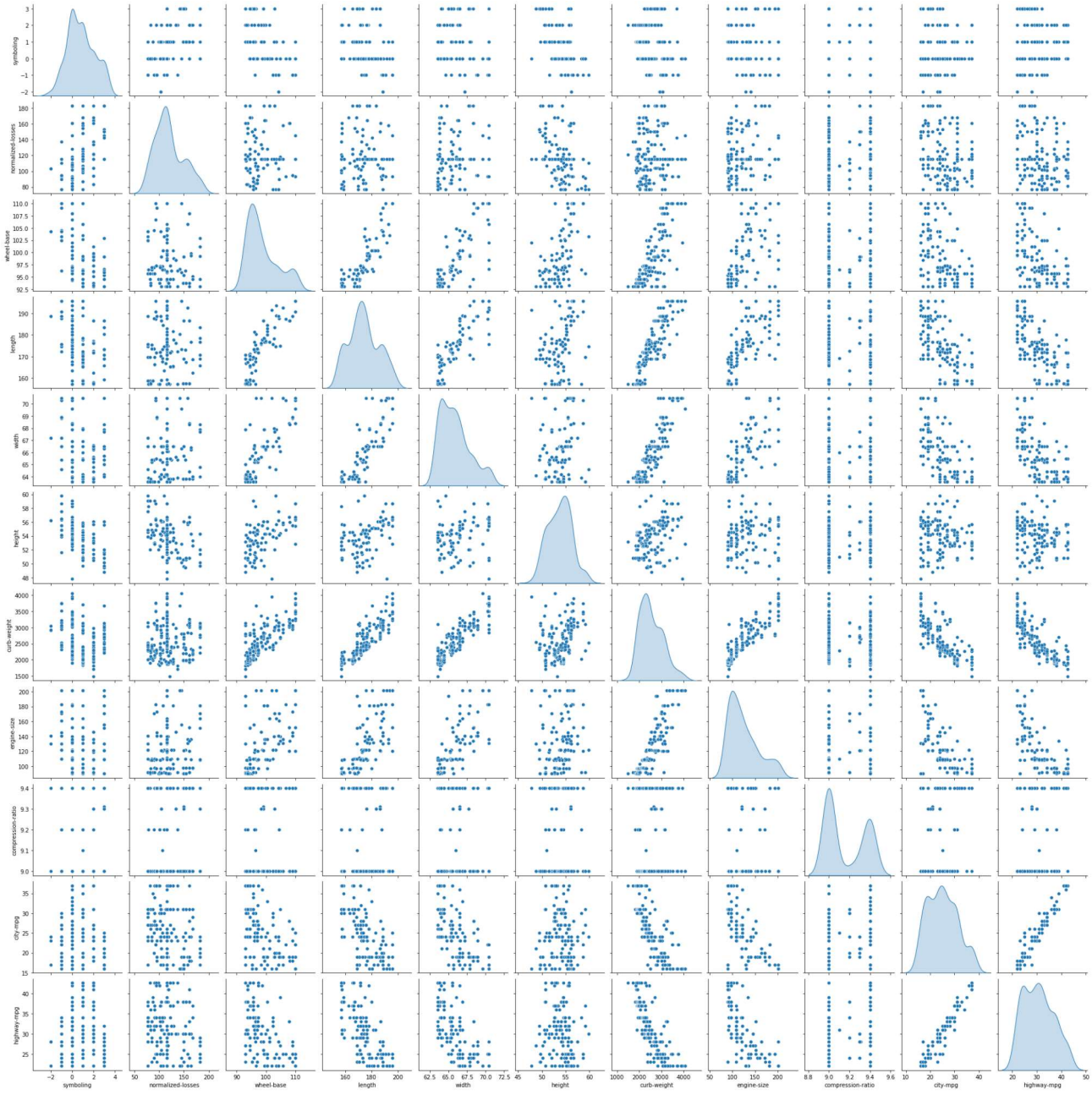
```
C:\Users\ap983\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing ot
her arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\ap983\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing ot
her arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\ap983\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing ot
her arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\ap983\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing ot
her arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\ap983\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing ot
her arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\ap983\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing ot
her arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\ap983\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing ot
her arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\ap983\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing ot
her arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\ap983\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing ot
her arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\ap983\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing ot
her arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\ap983\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing ot
her arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```


normalized-losses
Y-Axis
80 100 120 140 160 180
normalized-losses
symboling
Y-Axis
−2 −1 0 1 2 3
symboling
wheel-base
Y-Axis
92.5 95.0 97.5 100.0 102.5 105.0 107.5 110.0
wheel-base
length
Y-Axis
160 170 180 190
length
width
Y-Axis
64 65 66 67 68 69 70
width
height
Y-Axis
48 50 52 54 56 58 60
height
curb-weight
Y-Axis
1500 2000 2500 3000 3500 4000
curb-weight
engine-size
Y-Axis
100 120 140 160 180 200
engine-size
compression-ratio
Y-Axis
9.0 9.1 9.2 9.3 9.4
compression-ratio
city-mpg
Y-Axis
15 20 25 30 35
city-mpg
highway-mpg
Y-Axis
25 30 35 40
highway-mpg

Data Visualization

In [70]:
```
sns.pairplot(data,diag_kind='kde')
```

Out[70]:
```
<seaborn.axisgrid.PairGrid at 0x2053753aa30>
```



In [71]:
```
data.columns
```

Out[71]:
```
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')
```

In [72]:
```
data.describe(include='all').T
```

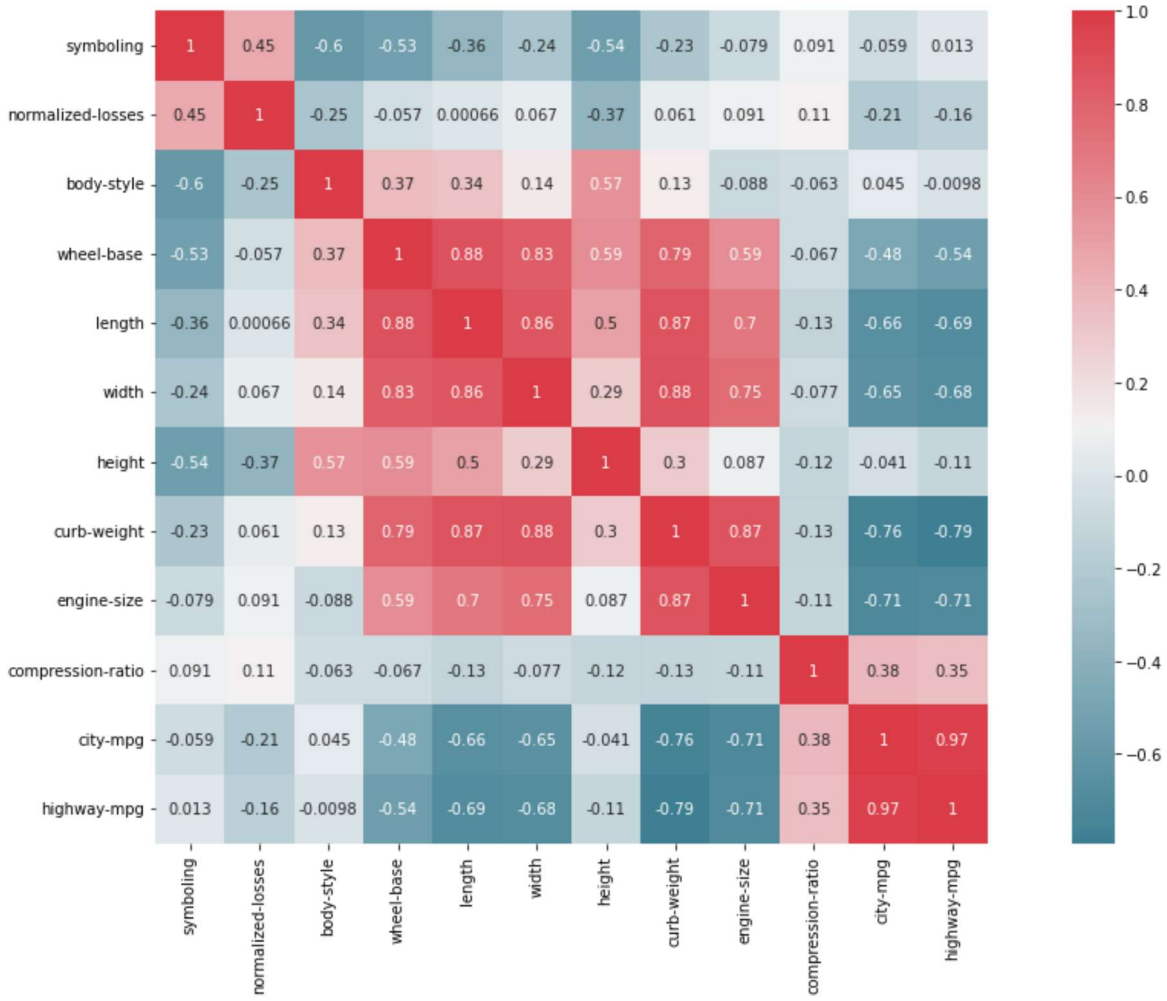
Out[72]:

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
symboling	205.0	NaN	NaN	NaN	0.834146	1.245307	-2.0	0.0	1.0	2.0	3.0
normalized-losses	205.0	NaN	NaN	NaN	119.959024	28.376086	77.2	101.0	115.0	137.0	182.4
make	205	22	toyota	32	NaN	NaN	NaN	NaN	NaN	NaN	NaN
fuel-type	205	2	gas	185	NaN	NaN	NaN	NaN	NaN	NaN	NaN
aspiration	205	2	std	168	NaN	NaN	NaN	NaN	NaN	NaN	NaN
num-of-doors	205	3	four	114	NaN	NaN	NaN	NaN	NaN	NaN	NaN
body-style	205	5	sedan	96	NaN	NaN	NaN	NaN	NaN	NaN	NaN
drive-wheels	205	3	fwd	120	NaN	NaN	NaN	NaN	NaN	NaN	NaN
engine-location	205	2	front	202	NaN	NaN	NaN	NaN	NaN	NaN	NaN
wheel-base	205.0	NaN	NaN	NaN	98.724488	5.154445	93.02	94.5	97.0	102.4	110.0
length	205.0	NaN	NaN	NaN	174.131805	11.193416	157.172	166.3	173.2	183.1	195.848
width	205.0	NaN	NaN	NaN	65.88761	1.961127	63.6	64.1	65.5	66.9	70.46
height	205.0	NaN	NaN	NaN	53.724878	2.443522	47.8	52.0	54.1	55.5	59.8
curb-weight	205.0	NaN	NaN	NaN	2555.565854	520.680204	1488.0	2145.0	2414.0	2935.0	4066.0
engine-type	205	7	ohc	148	NaN	NaN	NaN	NaN	NaN	NaN	NaN
num-of-cylinders	205	7	four	159	NaN	NaN	NaN	NaN	NaN	NaN	NaN
engine-size	205.0	NaN	NaN	NaN	124.81561	32.545885	90.0	97.0	120.0	141.0	201.2
fuel-system	205	8	mpfi	94	NaN	NaN	NaN	NaN	NaN	NaN	NaN
bore	205	39	3.62	23	NaN	NaN	NaN	NaN	NaN	NaN	NaN
stroke	205	37	3.4	20	NaN	NaN	NaN	NaN	NaN	NaN	NaN
compression-ratio	205.0	NaN	NaN	NaN	9.162488	0.187915	9.0	9.0	9.0	9.4	9.4
horsepower	205	60	68	19	NaN	NaN	NaN	NaN	NaN	NaN	NaN
peak-rpm	205	24	5500	37	NaN	NaN	NaN	NaN	NaN	NaN	NaN
city-mpg	205.0	NaN	NaN	NaN	25.087805	5.996903	16.0	19.0	24.0	30.0	37.0
highway-mpg	205.0	NaN	NaN	NaN	30.697756	6.047872	22.0	25.0	30.0	34.0	42.64
price	205	187	?	4	NaN	NaN	NaN	NaN	NaN	NaN	NaN

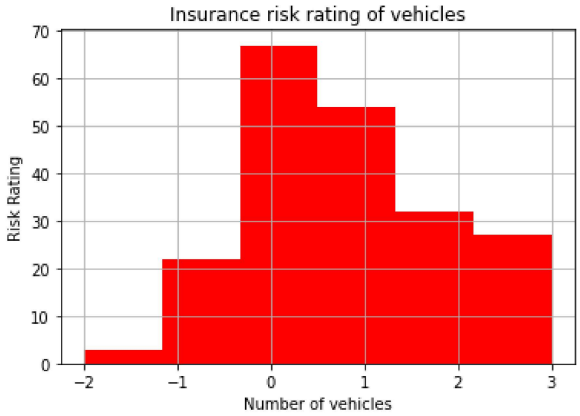
In [118]:
```
## Check the corelation using heatmap
f,ax = plt.subplots(figsize=(20,10))
corr = data.corr("pearson")
sns.heatmap(corr,mask=np.zeros_like(corr,dtype=np.bool),cmap=sns.diverging_palette(220,10,as_cmap=True),square
```

```
C:\Users\ap983\AppData\Local\Temp\ipykernel_16032\1233263262.py:4: DeprecationWarning: `np.bool` is a depreca
ted alias for the builtin `bool`. To silence this warning, use `bool` by itself. Doing this will not modify a
ny behavior and is safe. If you specifically wanted the numpy scalar type, use `np.bool_` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#
deprecations (https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations)
  sns.heatmap(corr,mask=np.zeros_like(corr,dtype=np.bool),cmap=sns.diverging_palette(220,10,as_cmap=True),squ
are=True,ax=ax,annot=True)
```

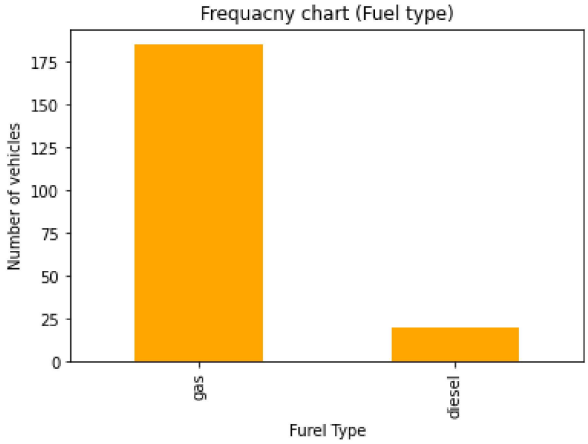
Out[118]: <AxesSubplot:>



In [119]:
```
## print the histogram
data.symboling.hist(bins=6,color='red');
plt.title('Insurance risk rating of vehicles')
plt.xlabel('Number of vehicles')
plt.ylabel('Risk Rating')
plt.show()
```



In [86]:
```
## Check the fuel Frequency  using plot method
data['fuel-type'].value_counts().plot(kind='bar',color='orange')
plt.title('Frequacny chart (Fuel type)')
plt.xlabel('Furel Type')
plt.ylabel('Number of vehicles');
```



In [87]:
```
data.head()
```

Out[87]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	com
0	3	115.0	alfa-romero	gas	std	two	convertible	rwd	front	93.02	...	130.0	mpfi	3.47	2.68	
1	3	115.0	alfa-romero	gas	std	two	convertible	rwd	front	93.02	...	130.0	mpfi	3.47	2.68	
2	1	115.0	alfa-romero	gas	std	two	hatchback	rwd	front	94.50	...	152.0	mpfi	2.68	3.47	
3	2	164.0	audi	gas	std	four	sedan	fwd	front	99.80	...	109.0	mpfi	3.19	3.4	
4	2	164.0	audi	gas	std	four	sedan	4wd	front	99.40	...	136.0	mpfi	3.19	3.4	

5 rows × 26 columns

Data Preprocessing

In [89]:
```
data['body-style'].head(20)
```

Out[89]:
```
0     convertible
1     convertible
2       hatchback
3           sedan
4           sedan
5           sedan
6           sedan
7           wagon
8           sedan
9       hatchback
10          sedan
11          sedan
12          sedan
13          sedan
14          sedan
15          sedan
16          sedan
17          sedan
18      hatchback
19      hatchback
Name: body-style, dtype: object
```

In [97]:
```
## Labele Encoding
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
data['body-style'] = labelencoder.fit_transform(data['body-style'])
```

In [98]:
```
## Data after enciding
data['body-style'].head(20)
```

Out[98]:
```
0     0
1     0
2     2
3     3
4     3
5     3
6     3
7     4
8     3
9     2
10    3
11    3
12    3
13    3
14    3
15    3
16    3
17    3
18    2
19    2
Name: body-style, dtype: int32
```

In [99]:
```
## Data before one hot Encoding
data['body-style'].head(10)
```

Out[99]:
```
0    0
1    0
2    2
3    3
4    3
5    3
6    3
7    4
8    3
9    2
Name: body-style, dtype: int32
```

In [103]:
```
## One hot Encoding
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder(handle_unknown='ignore')
enc_df=pd.DataFrame(enc.fit_transform(data[['body-style']]).toarray())
```

In [104]:
```
## Data after one hot endoding
enc_df
```

Out[104]:

	0	1	2	3	4
0	1.0	0.0	0.0	0.0	0.0
1	1.0	0.0	0.0	0.0	0.0
2	0.0	0.0	1.0	0.0	0.0
3	0.0	0.0	0.0	1.0	0.0
4	0.0	0.0	0.0	1.0	0.0
...	...	...	...	...	...
200	0.0	0.0	0.0	1.0	0.0
201	0.0	0.0	0.0	1.0	0.0
202	0.0	0.0	0.0	1.0	0.0
203	0.0	0.0	0.0	1.0	0.0
204	0.0	0.0	0.0	1.0	0.0

205 rows × 5 columns

In [ ]: