1-LIBRARIES,CONNECTING AND READING DATAFRAME

Import libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,r2_score
from sklearn.decomposition import PCA
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.decomposition import PCA
import re
import statsmodels.api as sm
from statsmodels.formula.api import ols




import warnings
warnings.filterwarnings('ignore')
```

Connecting data

```
df = pd.read_csv('/content/CarPrice_Assignment.csv')
```

Understanding Data

```
df.head(5)
```

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	c
0	1	3	alfa-romero giulia	gas	std	two	con
1	2	3	alfa-romero stelvio	gas	std	two	con
2	3	1	alfa-romero Quadrifoglio	gas	std	two	ha
3	4	2	audi 100 ls	gas	std	four	
4	5	2	audi 100ls	gas	std	four	

5 rows × 26 columns

```
df.tail(5)
```

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	car
200	201	-1	volvo 145e (sw)	gas	std	four	
201	202	-1	volvo 144ea	gas	turbo	four	
202	203	-1	volvo 244dl	gas	std	four	
203	204	-1	volvo 246	diesel	turbo	four	
204	205	-1	volvo 264gl	gas	turbo	four	

5 rows × 26 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   car_ID            205 non-null    int64  
 1   symboling         205 non-null    int64  
 2   CarName           205 non-null    object 
 3   fueltype          205 non-null    object 
 4   aspiration        205 non-null    object 
 5   doornumber        205 non-null    object 
 6   carbody           205 non-null    object 
 7   drivewheel        205 non-null    object 
 8   enginelocation    205 non-null    object 
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64  
 14  enginetype        205 non-null    object 
 15  cylindernumber    205 non-null    object 
 16  enginesize        205 non-null    int64  
 17  fuelsystem        205 non-null    object 
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64  
 22  peakrpm           205 non-null    int64  
 23  citympg           205 non-null    int64  
 24  highwaympg        205 non-null    int64  
 25  price             205 non-null    float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
```

```
df.shape
```

```
(205, 26)
```

```
df.dtypes
```

```
car_ID              int64
symboling           int64
CarName            object
fueltype           object
aspiration         object
doornumber         object
carbody            object
drivewheel         object
enginelocation     object
wheelbase         float64
carlength         float64
carwidth          float64
carheight         float64
curbweight          int64
enginetype         object
cylindernumber     object
enginesize          int64
fuelsystem         object
boreratio         float64
stroke            float64
compressionratio  float64
horsepower          int64
peakrpm             int64
citympg             int64
highwaympg          int64
price             float64
dtype: object
```

```
df.isnull().sum().sort_values(ascending=False)
```

```
car_ID              0
symboling           0
highwaympg          0
citympg             0
peakrpm             0
horsepower          0
compressionratio    0
stroke              0
boreratio           0
fuelsystem          0
enginesize          0
cylindernumber      0
enginetype          0
curbweight          0
carheight           0
carwidth            0
carlength           0
wheelbase           0
enginelocation      0
drivewheel          0
carbody             0
doornumber          0
aspiration          0
fueltype            0
CarName             0
price               0
dtype: int64
```

```
df[df.duplicated(keep=False)]
```

car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbo

0 rows × 26 columns

```
df.nunique()
```

```
car_ID              205
symboling             6
CarName             147
fueltype              2
aspiration            2
doornumber            2
carbody               5
drivewheel            3
enginelocation        2
wheelbase            53
carlength            75
carwidth             44
carheight            49
curbweight          171
enginetype            7
cylindernumber        7
enginesize           44
fuelsystem            8
boreratio            38
stroke               37
compressionratio     32
horsepower           59
peakrpm              23
citympg              29
highwaympg           30
price               189
dtype: int64
```

```
df.describe()
```

	car_ID	symboling	wheelbase	carlength	carwidth	carheigh
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.00000
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.72487
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.44352
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.80000
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.00000
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.10000
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.50000
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.80000

```
categorical_columns = ['fueltype','aspiration','doornumber','carbody','
    'cylindernumber',
    'fuelsystem'
]

for col in categorical_columns:

    print(f"Category in {col} is : {df[col].unique()}")
```

```
Category in fueltype is : ['gas' 'diesel']
Category in aspiration is : ['std' 'turbo']
Category in doornumber is : ['two' 'four']
Category in carbody is : ['convertible' 'hatchback' 'sedan' 'wagon'
Category in drivewheel is : ['rwd' 'fwd' '4wd']
Category in enginelocation is : ['front' 'rear']
Category in enginetype is : ['dohc' 'ohcv' 'ohc' 'l' 'rotor' 'ohcf'
Category in cylindernumber is : ['four' 'six' 'five' 'three' 'twelve
Category in fuelsystem is : ['mpfi' '2bbl' 'mfi' '1bbl' 'spfi' '4bbl
```

Observations:

1. **Car Variety under Car_ID column :**
    - Dataset covers 205 car models, each unique in attributes.
    - Sizes, weights, and engine capacities vary across cars.
2. **Price Range under Price column:**
    - Mini Price $5,118
    - Maxi Price $45,400
    - Average car price is about $13,276.
3. **Engine Diversity :**
    - Engines differ in bore diameter (`boreratio`) and stroke length (`stroke`).
    - Risk ratings (`symboling`) vary from -2 to 3.
4. **Data Distribution :**
    - Features like `carwidth`, `curbweight`, `enginesize` show notable variability.
    - Potential outliers present in `compressionratio` and `horsepower`.

2- DATA CLEANING

Dealing with Datatypes

```
df.dtypes
```

```
car_ID              int64
symboling           int64
CarName            object
fueltype           object
aspiration         object
doornumber         object
carbody            object
drivewheel         object
enginelocation     object
wheelbase         float64
carlength         float64
carwidth          float64
carheight         float64
curbweight          int64
enginetype         object
cylindernumber     object
enginesize          int64
fuelsystem         object
boreratio         float64
stroke            float64
compressionratio  float64
horsepower          int64
peakrpm             int64
citympg             int64
highwaympg          int64
price             float64
dtype: object
```

Dealing with Special Keys

```
# Define the special characters pattern
special_characters = r'[!@#$%^&*()_+{}\[\]:;<>,.?~\\|"]'

def highlight_special_characters(val):
    if isinstance(val, str) and re.search(special_characters, val):
        return 'background-color: blue'
```

```
    return ''

styled_data = df.style.applymap(highlight_special_characters)
styled_data
```

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	
0	1	3	alfa-romero giulia	gas	std	two	c
1	2	3	alfa-romero stelvio	gas	std	two	c
2	3	1	alfa-romero Quadrifoglio	gas	std	two	
3	4	2	audi 100 ls	gas	std	four	
4	5	2	audi 100ls	gas	std	four	
5	6	2	audi fox	gas	std	two	
6	7	1	audi 100ls	gas	std	four	
7	8	1	audi 5000	gas	std	four	
8	9	1	audi 4000	gas	turbo	four	
9	10	0	audi 5000s (diesel)	gas	turbo	two	
10	11	2	bmw 320i	gas	std	two	
11	12	0	bmw 320i	gas	std	four	
12	13	0	bmw x1	gas	std	two	
13	14	0	bmw x3	gas	std	four	
14	15	1	bmw z4	gas	std	four	
15	16	0	bmw x4	gas	std	four	
16	17	0	bmw x5	gas	std	two	
17	18	0	bmw x3	gas	std	four	
18	19	2	chevrolet impala	gas	std	two	
19	20	1	chevrolet monte carlo	gas	std	two	
20	21	0	chevrolet vega 2300	gas	std	four	

21	22	1	dodge rampage	gas	std	two
22	23	1	dodge challenger se	gas	std	two
23	24	1	dodge d200	gas	turbo	two
24	25	1	dodge monaco (sw)	gas	std	four
25	26	1	dodge colt hardtop	gas	std	four
26	27	1	dodge colt (sw)	gas	std	four
27	28	1	dodge coronet custom	gas	turbo	two
28	29	-1	dodge dart custom	gas	std	four
29	30	3	dodge coronet custom (sw)	gas	turbo	two
30	31	2	honda civic	gas	std	two
31	32	2	honda civic cvcc	gas	std	two
32	33	1	honda civic	gas	std	two
33	34	1	honda accord cvcc	gas	std	two
34	35	1	honda civic cvcc	gas	std	two
35	36	0	honda accord lx	gas	std	four
36	37	0	honda civic 1500 gl	gas	std	four
37	38	0	honda accord	gas	std	two
38	39	0	honda civic 1300	gas	std	two

39	40	0	honda prelude	gas	std	four
40	41	0	honda accord	gas	std	four
41	42	0	honda civic	gas	std	four
42	43	1	honda civic (auto)	gas	std	two
43	44	0	isuzu MU-X	gas	std	four
44	45	1	isuzu D-Max	gas	std	two
45	46	0	isuzu D-Max V-Cross	gas	std	four
46	47	2	isuzu D-Max	gas	std	two
47	48	0	jaguar xj	gas	std	four
48	49	0	jaguar xf	gas	std	four
49	50	0	jaguar xk	gas	std	two
50	51	1	maxda rx3	gas	std	two
51	52	1	maxda glc deluxe	gas	std	two
52	53	1	mazda rx2 coupe	gas	std	two
53	54	1	mazda rx-4	gas	std	four
54	55	1	mazda glc deluxe	gas	std	four
55	56	3	mazda 626	gas	std	two
56	57	3	mazda glc	gas	std	two
57	58	3	mazda rx-7 gs	gas	std	two
58	59	3	mazda glc 4	gas	std	two
59	60	1	mazda 626	gas	std	two
60	61	0	mazda glc custom l	gas	std	four

61	62	1	mazda glc custom	gas	std	two
62	63	0	mazda rx-4	gas	std	four
63	64	0	mazda glc deluxe	diesel	std	four
64	65	0	mazda 626	gas	std	four
65	66	0	mazda glc	gas	std	four
66	67	0	mazda rx-7 gs	diesel	std	four
67	68	-1	buick electra 225 custom	diesel	turbo	four
68	69	-1	buick century luxus (sw)	diesel	turbo	four
69	70	0	buick century	diesel	turbo	two
70	71	-1	buick skyhawk	diesel	turbo	four
71	72	-1	buick opel isuzu deluxe	gas	std	four
72	73	3	buick skylark	gas	std	two
73	74	0	buick century special	gas	std	four
74	75	1	buick regal sport coupe (turbo)	gas	std	two
75	76	1	mercury cougar	gas	turbo	two
76	77	2	mitsubishi mirage	gas	std	two
77	78	2	mitsubishi lancer	gas	std	two
78	79	2	mitsubishi	gas	std	two

78	79	2	outlander	gas	std	two
79	80	1	mitsubishi g4	gas	turbo	two
80	81	3	mitsubishi mirage g4	gas	turbo	two
81	82	3	mitsubishi g4	gas	std	two
82	83	3	mitsubishi outlander	gas	turbo	two
83	84	3	mitsubishi g4	gas	turbo	two
84	85	3	mitsubishi mirage g4	gas	turbo	two
85	86	1	mitsubishi montero	gas	std	four
86	87	1	mitsubishi pajero	gas	std	four
87	88	1	mitsubishi outlander	gas	turbo	four
88	89	-1	mitsubishi mirage g4	gas	std	four
89	90	1	Nissan versa	gas	std	two
90	91	1	nissan gt-r	diesel	std	two
91	92	1	nissan rogue	gas	std	two
92	93	1	nissan latio	gas	std	four
93	94	1	nissan titan	gas	std	four
94	95	1	nissan leaf	gas	std	two
95	96	1	nissan juke	gas	std	two
96	97	1	nissan latio	gas	std	four
97	98	1	nissan note	gas	std	four
98	99	2	nissan clipper	gas	std	two
			nissan			

99	100	0	nissan rogue	gas	std	four
100	101	0	nissan nv200	gas	std	four
101	102	0	nissan dayz	gas	std	four
102	103	0	nissan fuga	gas	std	four
103	104	0	nissan otti	gas	std	four
104	105	3	nissan teana	gas	std	two
105	106	3	nissan kicks	gas	turbo	two
106	107	1	nissan clipper	gas	std	two
107	108	0	peugeot 504	gas	std	four
108	109	0	peugeot 304	diesel	turbo	four
109	110	0	peugeot 504 (sw)	gas	std	four
110	111	0	peugeot 504	diesel	turbo	four
111	112	0	peugeot 504	gas	std	four
112	113	0	peugeot 604sl	diesel	turbo	four
113	114	0	peugeot 504	gas	std	four
114	115	0	peugeot 505s turbo diesel	diesel	turbo	four
115	116	0	peugeot 504	gas	std	four
116	117	0	peugeot 504	diesel	turbo	four
117	118	0	peugeot 604sl	gas	turbo	four
118	119	1	plymouth fury iii	gas	std	two

			fury iii			
119	120	1	plymouth cricket	gas	turbo	two
120	121	1	plymouth fury iii	gas	std	four
121	122	1	plymouth satellite custom (sw)	gas	std	four
122	123	1	plymouth fury gran sedan	gas	std	four
123	124	-1	plymouth valiant	gas	std	four
124	125	3	plymouth duster	gas	turbo	two
125	126	3	porsche macan	gas	std	two
126	127	3	porcshce panamera	gas	std	two
127	128	3	porsche cayenne	gas	std	two
128	129	3	porsche boxter	gas	std	two
129	130	1	porsche cayenne	gas	std	two
130	131	0	renault 12tl	gas	std	four
131	132	2	renault 5 gtl	gas	std	two
132	133	3	saab 99e	gas	std	two
133	134	2	saab 99le	gas	std	four
134	135	3	saab 99le	gas	std	two
135	136	2	saab 99gle	gas	std	four
136	137	3	saab 99gle	gas	turbo	two
137	138	2	saab 99e	gas	turbo	four
138	139	2	subaru	gas	std	two
139	140	2	subaru dl	gas	std	two

140	141	2	subaru dl	gas	std	two
141	142	0	subaru	gas	std	four
142	143	0	subaru brz	gas	std	four
143	144	0	subaru baja	gas	std	four
144	145	0	subaru r1	gas	std	four
145	146	0	subaru r2	gas	turbo	four
146	147	0	subaru trezia	gas	std	four
147	148	0	subaru tribeca	gas	std	four
148	149	0	subaru dl	gas	std	four
149	150	0	subaru dl	gas	turbo	four
150	151	1	toyota corona mark ii	gas	std	two
151	152	1	toyota corona	gas	std	two
152	153	1	toyota corolla 1200	gas	std	four
153	154	0	toyota corona hardtop	gas	std	four
154	155	0	toyota corolla 1600 (sw)	gas	std	four
155	156	0	toyota carina	gas	std	four
156	157	0	toyota mark ii	gas	std	four
157	158	0	toyota corolla 1200	gas	std	four
158	159	0	toyota corona	diesel	std	four
			toyota			

159	160	0	toyota corolla	diesel	std	four
160	161	0	toyota corona	gas	std	four
161	162	0	toyota corolla	gas	std	four
162	163	0	toyota mark ii	gas	std	four
163	164	1	toyota corolla liftback	gas	std	two
164	165	1	toyota corona	gas	std	two
165	166	1	toyota celica gt liftback	gas	std	two
166	167	1	toyota corolla tercel	gas	std	two
167	168	2	toyota corona liftback	gas	std	two
168	169	2	toyota corolla	gas	std	two
169	170	2	toyota starlet	gas	std	two
170	171	2	toyota tercel	gas	std	two
171	172	2	toyota corolla	gas	std	two
172	173	2	toyota cressida	gas	std	two
173	174	-1	toyota corolla	gas	std	four
174	175	-1	toyota celica gt	diesel	turbo	four
175	176	-1	toyota corona	gas	std	four

176	177	-1	toyota corolla	gas	std	four	
177	178	-1	toyota mark ii	gas	std	four	
178	179	3	toyota corolla liftback	gas	std	two	
179	180	3	toyota corona	gas	std	two	
180	181	-1	toyota starlet	gas	std	four	
181	182	-1	toyouta tercel	gas	std	four	
182	183	2	vokswagen rabbit	diesel	std	two	
183	184	2	volkswagen 1131 deluxe sedan	gas	std	two	
184	185	2	volkswagen model 111	diesel	std	four	
185	186	2	volkswagen type 3	gas	std	four	
186	187	2	volkswagen 411 (sw)	gas	std	four	
187	188	2	volkswagen super beetle	diesel	turbo	four	
188	189	2	volkswagen dasher	gas	std	four	
189	190	3	vw dasher	gas	std	two	d
190	191	3	vw rabbit	gas	std	two	
191	192	0	volkswagen rabbit	gas	std	four	
192	193	0	volkswagen rabbit custom	diesel	turbo	four	
193	194	0	volkswagen dasher	gas	std	four	

194	195	-2	volvo 145e (sw)	gas	std	four
195	196	-1	volvo 144ea	gas	std	four
196	197	-2	volvo 244dl	gas	std	four
197	198	-1	volvo 245	gas	std	four
198	199	-2	volvo 264gl	gas	turbo	four
199	200	-1	volvo diesel	gas	turbo	four
200	201	-1	volvo 145e (sw)	gas	std	four
201	202	-1	volvo 144ea	gas	turbo	four
202	203	-1	volvo 244dl	gas	std	four
203	204	-1	volvo 246	diesel	turbo	four
204	205	-1	volvo 264gl	gas	turbo	four

Dealing with Duplicated values

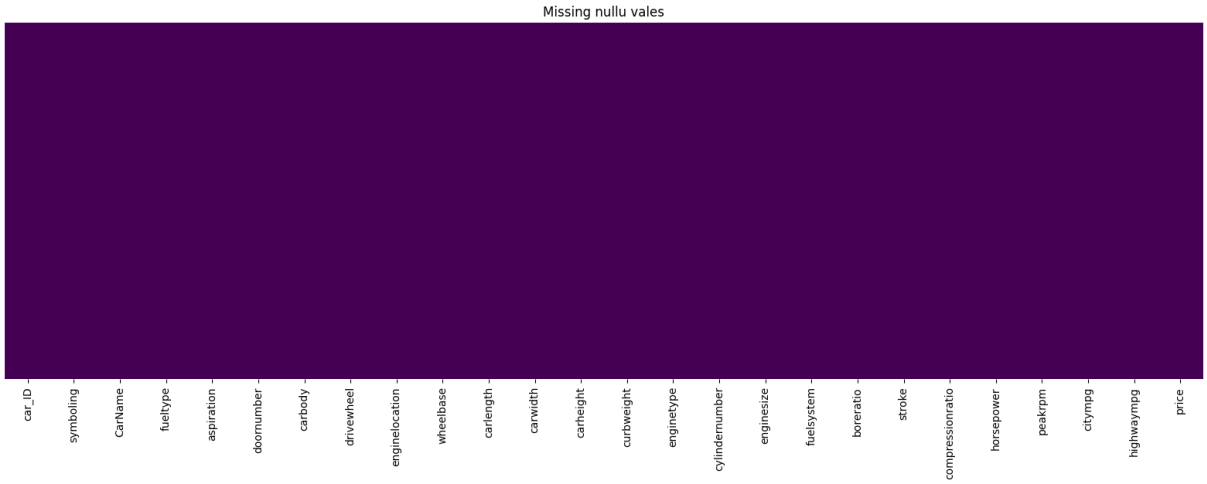
```
df.duplicated()
```

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

Dealing with missing values

```
import matplotlib
matplotlib.rcParams['figure.figsize'] = (20,6)
sns.heatmap(df.isnull(),yticklabels = False, cbar = False , cmap = 'vir
plt.title("Missing nullu vales")
```

```
Text(0.5, 1.0, 'Missing nullu vales')
```



Observation:

1) Dealing with Datatypes: No need to convert any datatypes

2) Dealing with Special keys: Found special values, but not affecting any issue for future analysis

3) Dealing with Duplicated values: No duplicates were found

4)Dealing with missing values: No missing value found

3-EXPLORATORY DATA ANALYSIS

Statistics Summary

```
df.describe().T
```

	count	mean	std	min	25%	50
car_ID	205.0	103.000000	59.322565	1.00	52.00	103.0
symboling	205.0	0.834146	1.245307	-2.00	0.00	1.0
wheelbase	205.0	98.756585	6.021776	86.60	94.50	97.0
carlength	205.0	174.049268	12.337289	141.10	166.30	173.2
carwidth	205.0	65.907805	2.145204	60.30	64.10	65.5
carheight	205.0	53.724878	2.443522	47.80	52.00	54.1
curbweight	205.0	2555.565854	520.680204	1488.00	2145.00	2414.0
enginesize	205.0	126.907317	41.642693	61.00	97.00	120.0
boreratio	205.0	3.329756	0.270844	2.54	3.15	3.3
stroke	205.0	3.255415	0.313597	2.07	3.11	3.2
compressionratio	205.0	10.142537	3.972040	7.00	8.60	9.0
horsepower	205.0	104.117073	39.544167	48.00	70.00	95.0
peakrpm	205.0	5125.121951	476.985643	4150.00	4800.00	5200.0
citympg	205.0	25.219512	6.542142	13.00	19.00	24.0
highwaympg	205.0	30.751220	6.886443	16.00	25.00	30.0
price	205.0	13276.710571	7988.852332	5118.00	7788.00	10295.0

```
df.describe(include='all').T
```

	count	unique	top	freq	mean	std
car_ID	205.0	NaN	NaN	NaN	103.0	59.322565
symboling	205.0	NaN	NaN	NaN	0.834146	1.245307
CarName	205	147	toyota corona	6	NaN	NaN
fueltype	205	2	gas	185	NaN	NaN
aspiration	205	2	std	168	NaN	NaN
doornumber	205	2	four	115	NaN	NaN
carbody	205	5	sedan	96	NaN	NaN

drivewheel	205	3	fwd	120	NaN	NaN	
enginelocation	205	2	front	202	NaN	NaN	
wheelbase	205.0	NaN	NaN	NaN	98.756585	6.021776	
carlength	205.0	NaN	NaN	NaN	174.049268	12.337289	
carwidth	205.0	NaN	NaN	NaN	65.907805	2.145204	
carheight	205.0	NaN	NaN	NaN	53.724878	2.443522	
curbweight	205.0	NaN	NaN	NaN	2555.565854	520.680204	1
enginetype	205	7	ohc	148	NaN	NaN	
cylindernumber	205	7	four	159	NaN	NaN	
enginesize	205.0	NaN	NaN	NaN	126.907317	41.642693	
fuelsystem	205	8	mpfi	94	NaN	NaN	
boreratio	205.0	NaN	NaN	NaN	3.329756	0.270844	
stroke	205.0	NaN	NaN	NaN	3.255415	0.313597	
compressionratio	205.0	NaN	NaN	NaN	10.142537	3.97204	
horsepower	205.0	NaN	NaN	NaN	104.117073	39.544167	
peakrpm	205.0	NaN	NaN	NaN	5125.121951	476.985643	4
citympg	205.0	NaN	NaN	NaN	25.219512	6.542142	
highwaympg	205.0	NaN	NaN	NaN	30.75122	6.886443	
price	205.0	NaN	NaN	NaN	13276.710571	7988.852332	5

```
cat_cols=df.select_dtypes(include=['object']).columns
num_cols = df.select_dtypes(include=np.number).columns.tolist()
print("Categorical Variables:")
print(cat_cols)
print("Numerical Variables:")
print(num_cols)
```

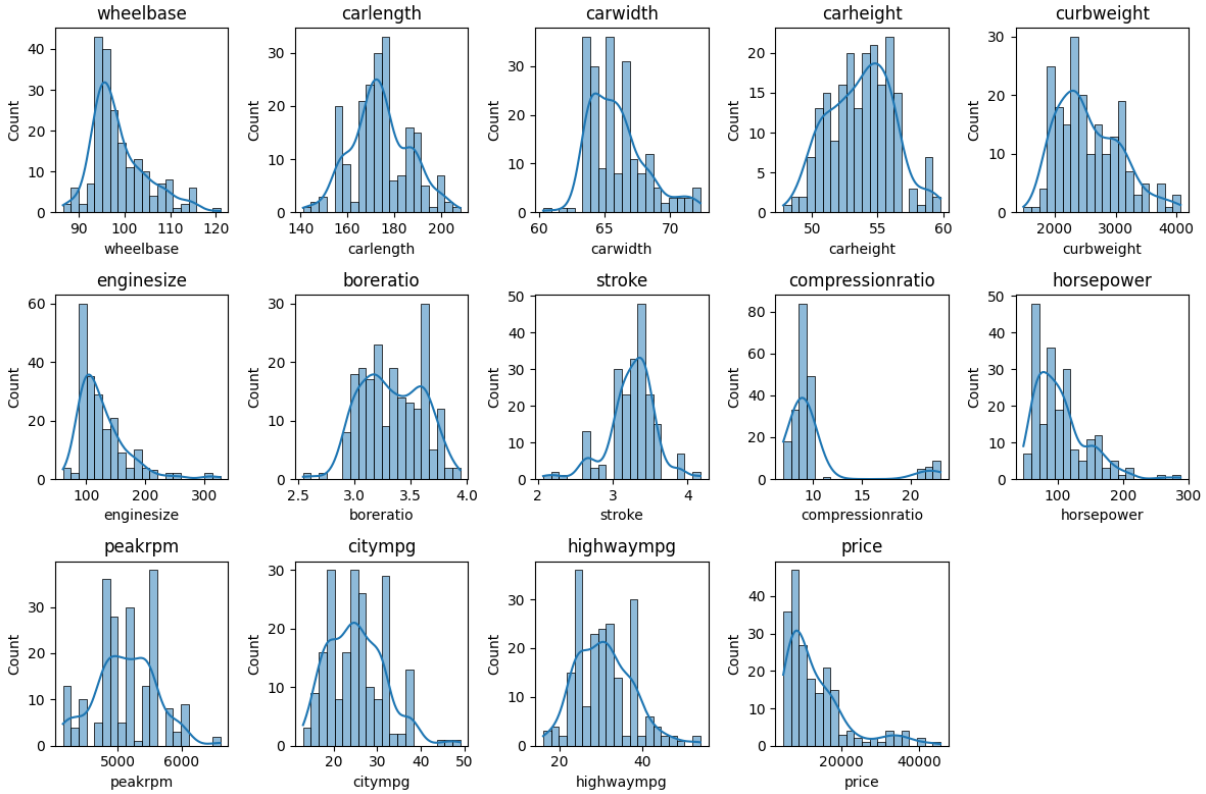
```
Categorical Variables:
Index(['CarName', 'fueltype', 'aspiration', 'doornumber', 'carbody',
       'drivewheel', 'enginelocation', 'enginetype', 'cylindernumber
       'fuelsystem'],
      dtype='object')
Numerical Variables:
['car_ID', 'symboling', 'wheelbase', 'carlength', 'carwidth', 'carhe
```

Observation:

Univariate Analysis

NUMERIC VARIABLES

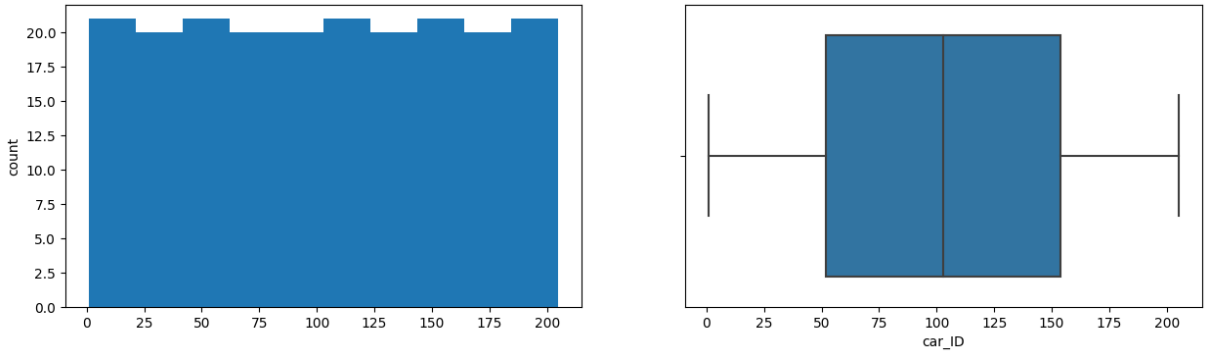
```
# Distribution of Numerical Features
numerical_features = ['wheelbase', 'carlength', 'carwidth', 'carheight'
                      'enginesize', 'boreratio', 'stroke', 'compression
                      'peakrpm', 'citympg', 'highwaympg', 'price']

plt.figure(figsize=(12, 8))
for feature in numerical_features:
    plt.subplot(3, 5, numerical_features.index(feature) + 1)
    sns.histplot(data=df[feature], bins=20, kde=True)
    plt.title(feature)
plt.tight_layout()
plt.show()
```



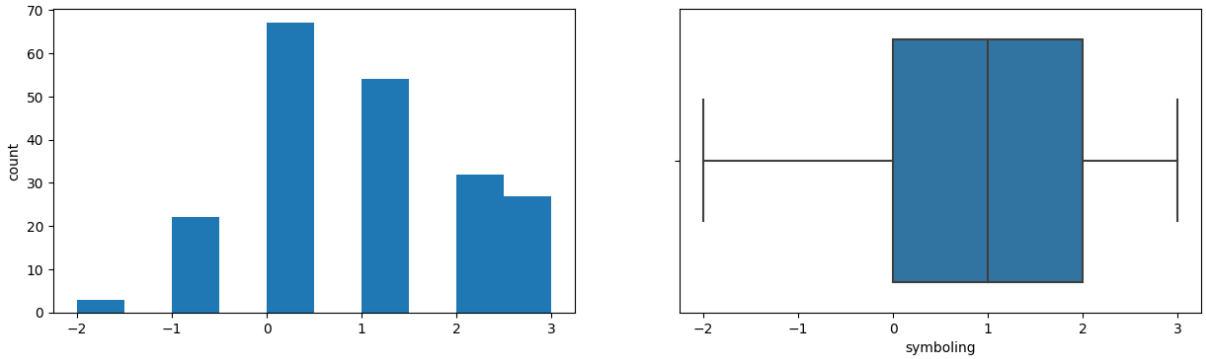

```
for col in num_cols:
    print(col)
    print('Skew :', round(df[col].skew(), 2))
    plt.figure(figsize = (15, 4))
```

```
    plt.subplot(1, 2, 1)
    df[col].hist(grid=False)
    plt.ylabel('count')
    plt.subplot(1, 2, 2)
    sns.boxplot(x=df[col])
    plt.show()
```

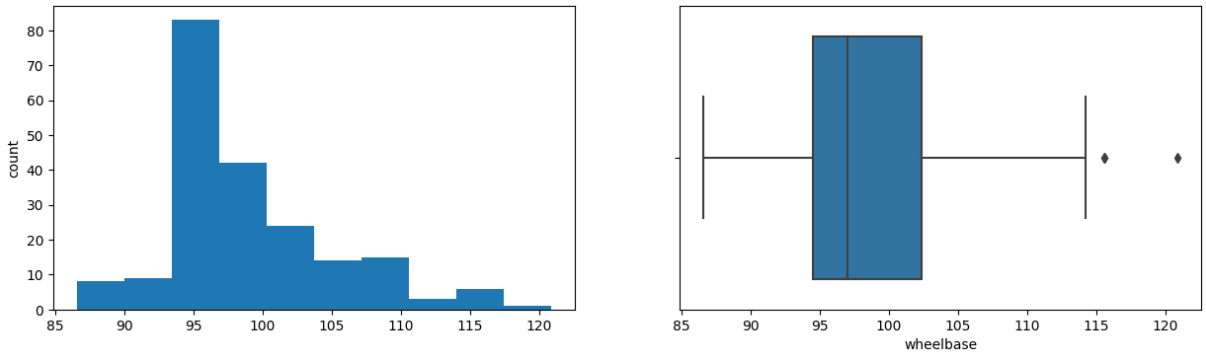
car_ID
Skew : 0.0



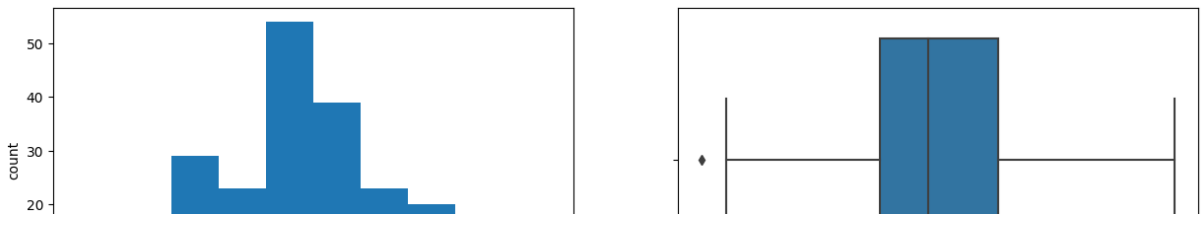
symboling
Skew : 0.21

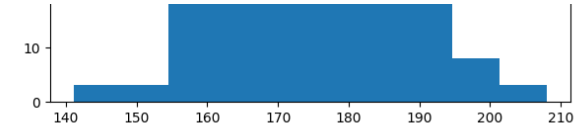
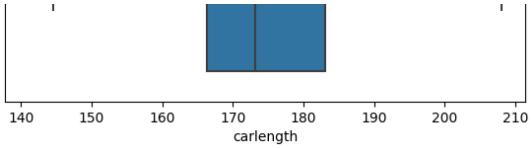


wheelbase
Skew : 1.05



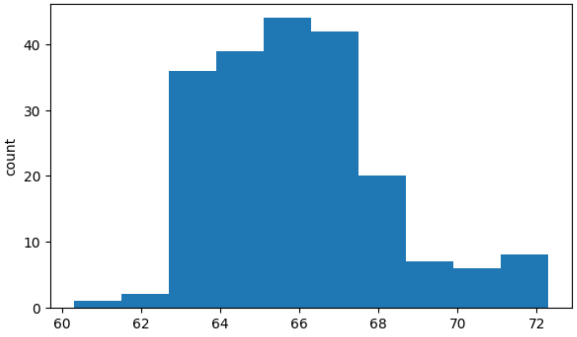
carlength
Skew : 0.16

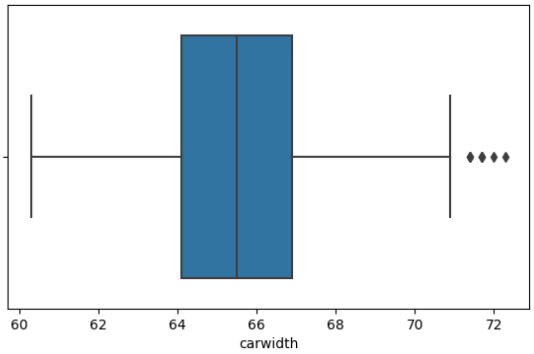


carwidth

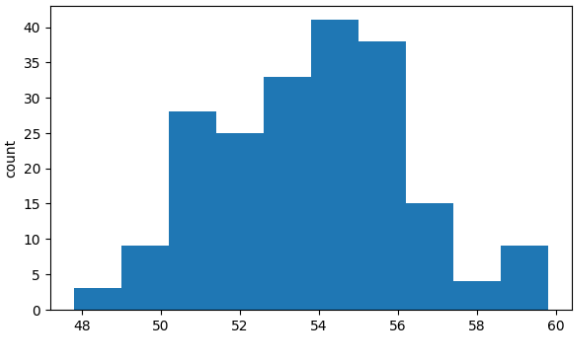
Skew : 0.9

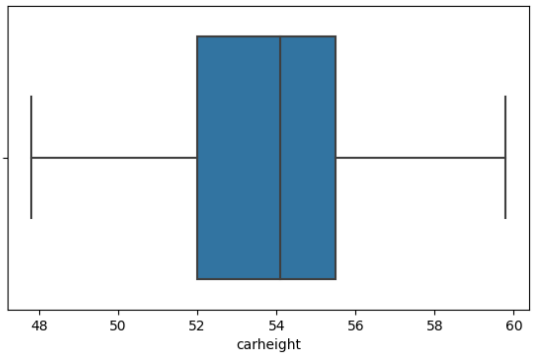




carheight

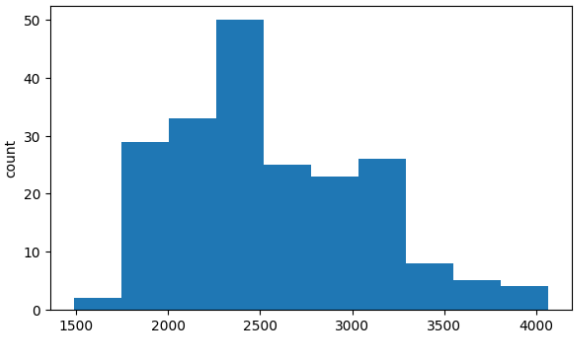
Skew : 0.06

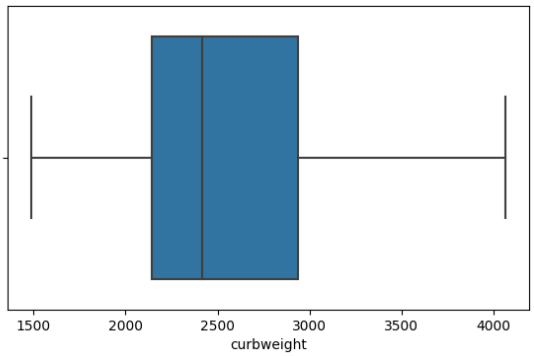




curbweight

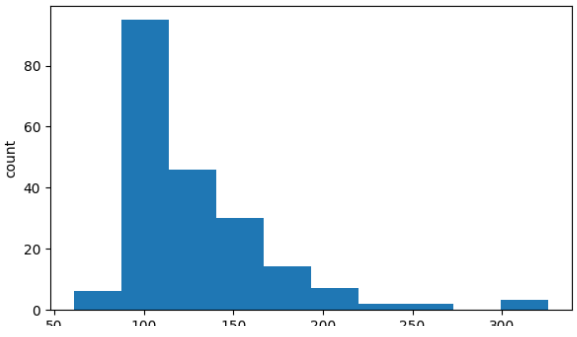
Skew : 0.68

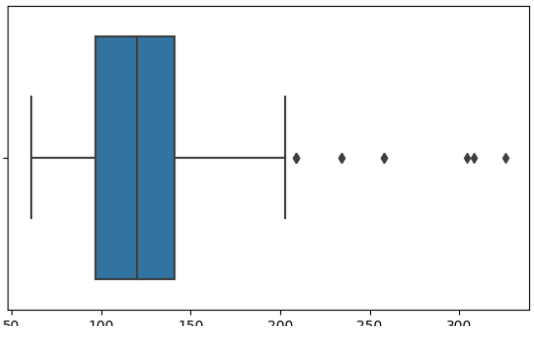




enginesize

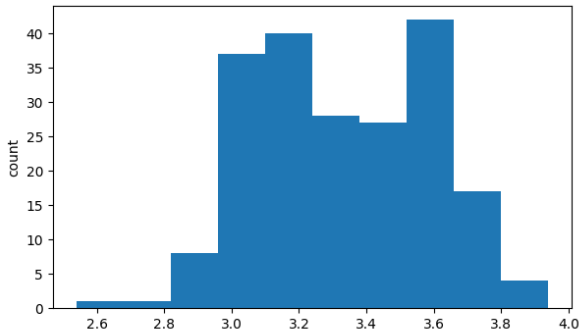
Skew : 1.95

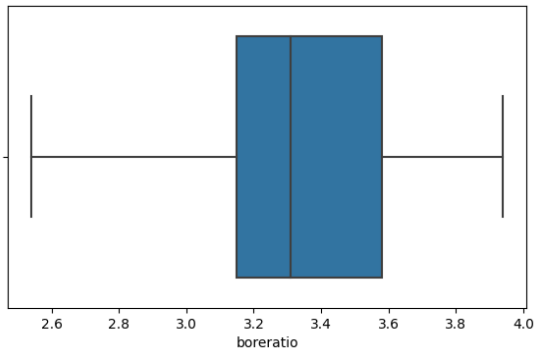




boreratio

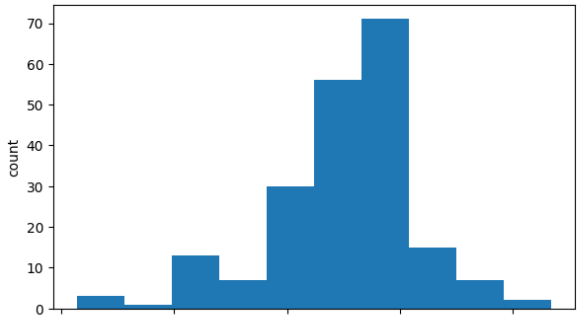
Skew : 0.02

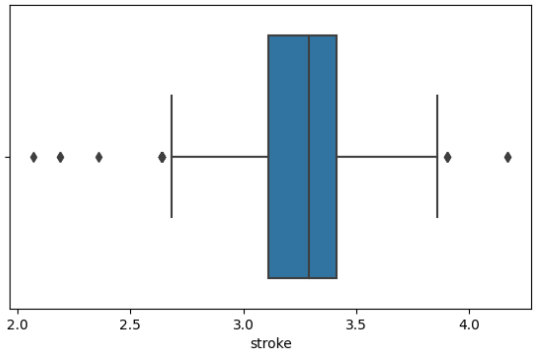




stroke

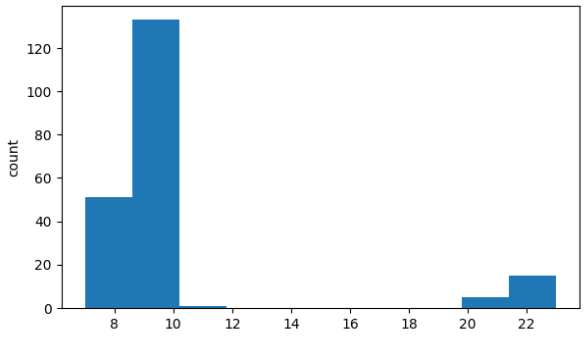
Skew : -0.69

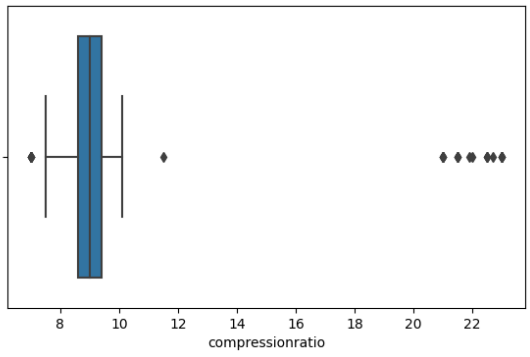




compressionratio

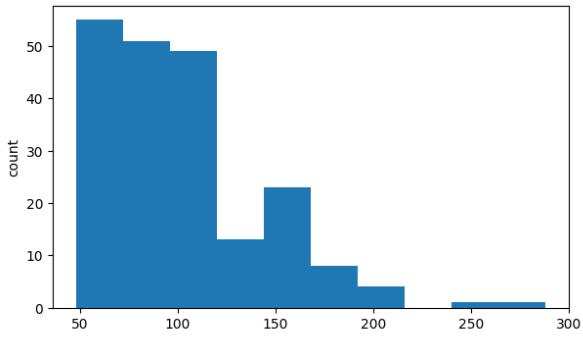
Skew : 2.61

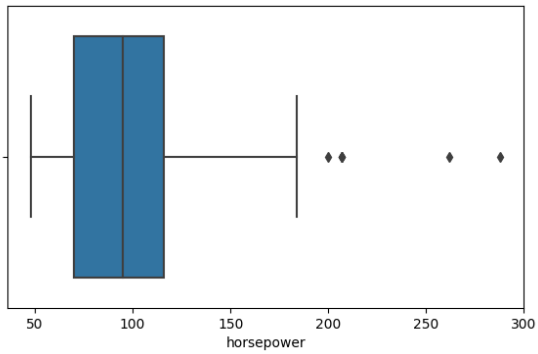




horsepower

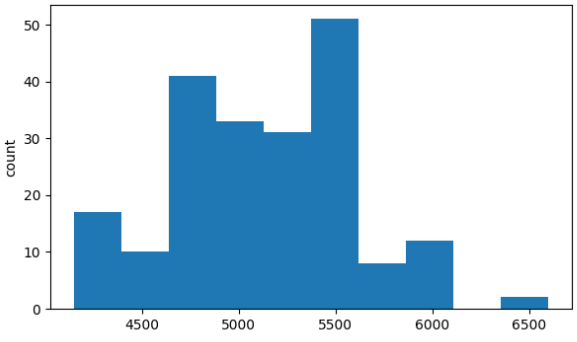
Skew : 1.41

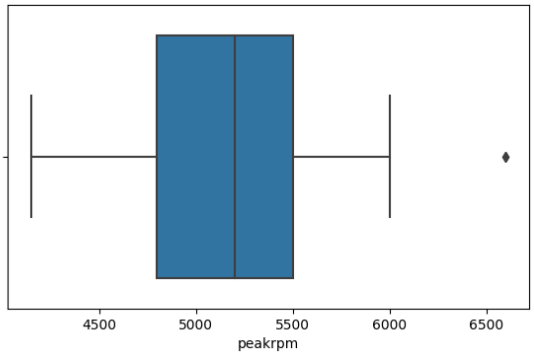




peakrpm

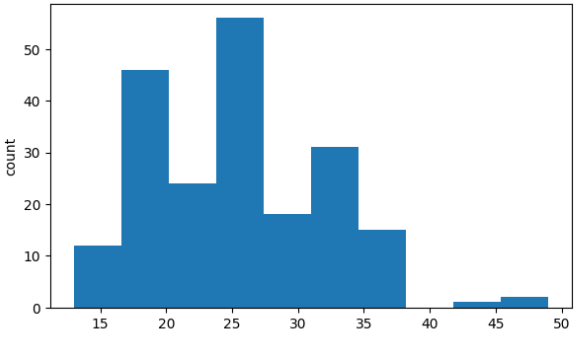
Skew : 0.08

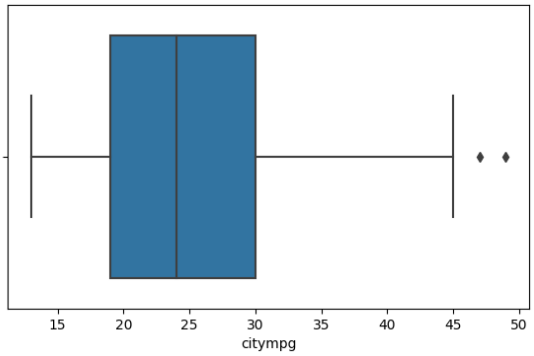




citympg

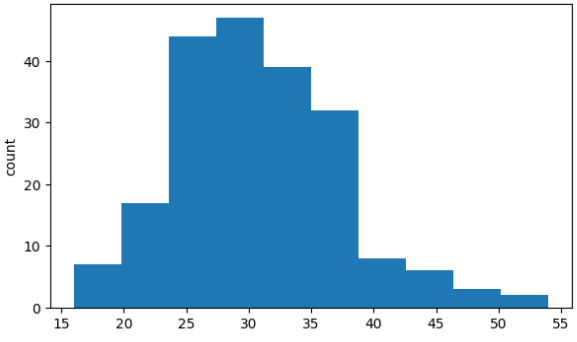
Skew : 0.66

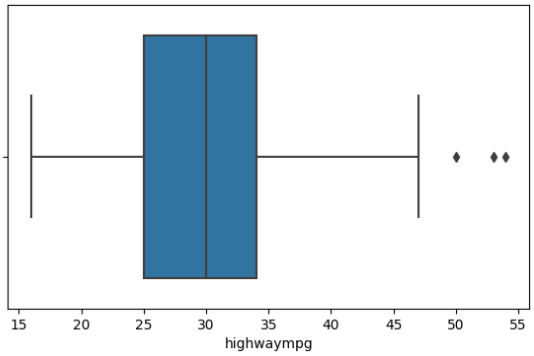




highwaympg

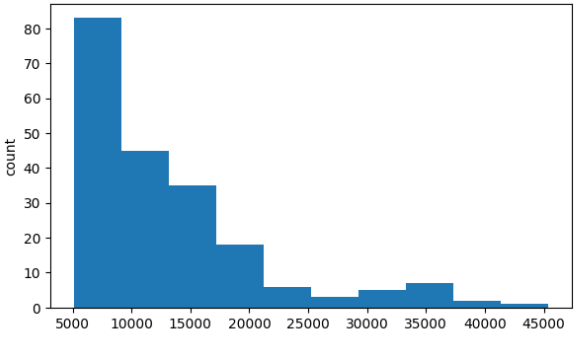
Skew : 0.54

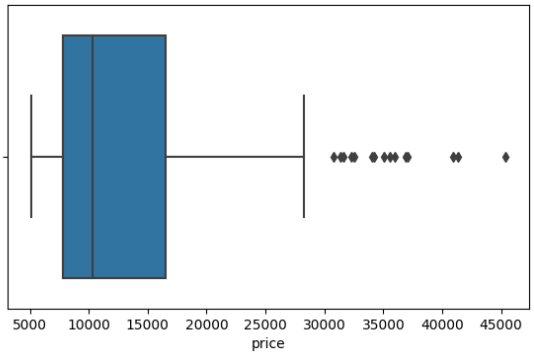




price

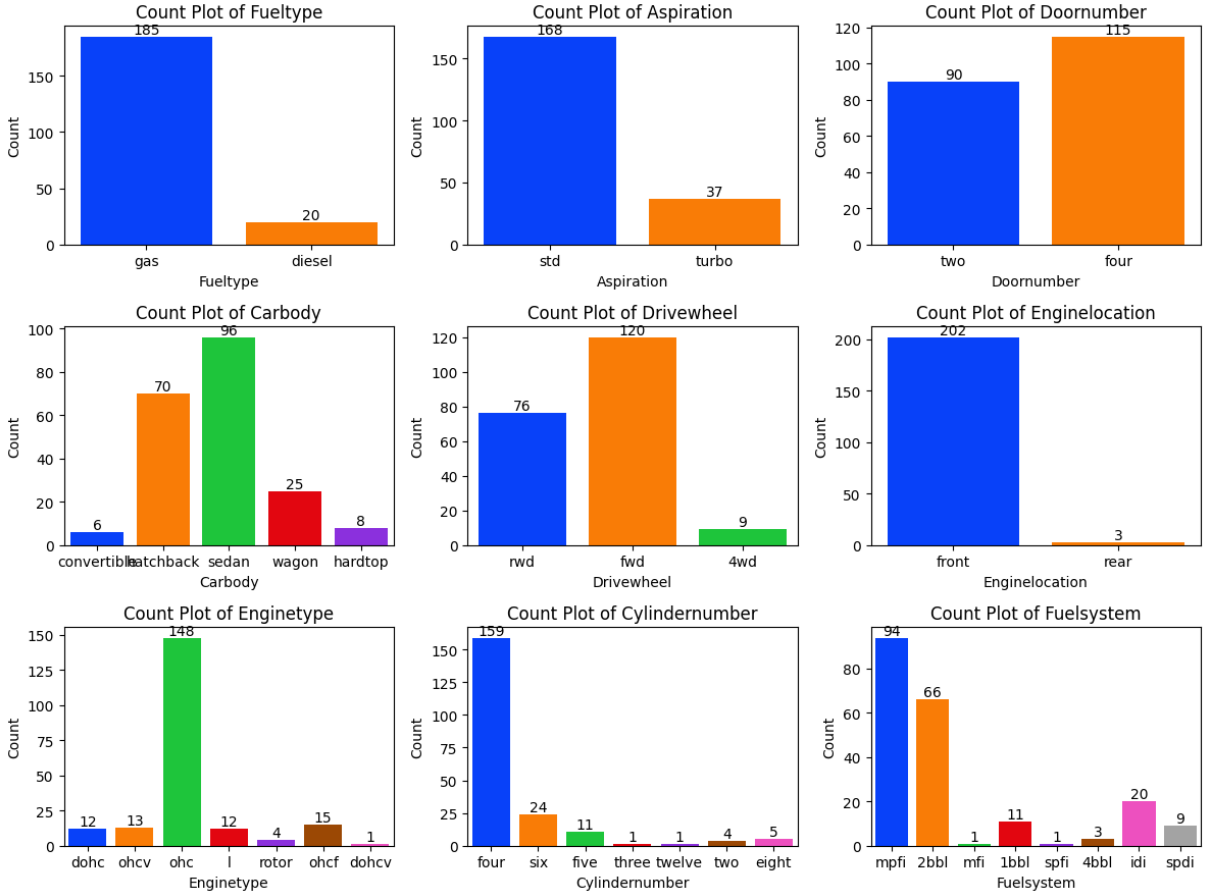
Skew : 1.78





CATEGORICAL VARIABLES

```
# Define the list of categorical columns to analyze
categorical_columns = ['fueltype', 'aspiration', 'doornumber', 'carbody
                       'enginelocation', 'enginetype', 'cylindernumber'

# Create subplots
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(12, 9))
axes = axes.ravel()  # Flatten the 2D array of axes

# Loop through each categorical column
for i, column in enumerate(categorical_columns):
    sns.countplot(x=df[column], data=df, palette='bright', ax=axes[i], s
    for container in axes[i].containers:
        axes[i].bar_label(container, color='black', size=10)
    axes[i].set_title(f'Count Plot of {column.capitalize()}')
    axes[i].set_xlabel(column.capitalize())
    axes[i].set_ylabel('Count')

# Adjust layout and show plots
plt.tight_layout()
plt.show()
```



Observation:

Outliers found in numerical variables

Data Transformation

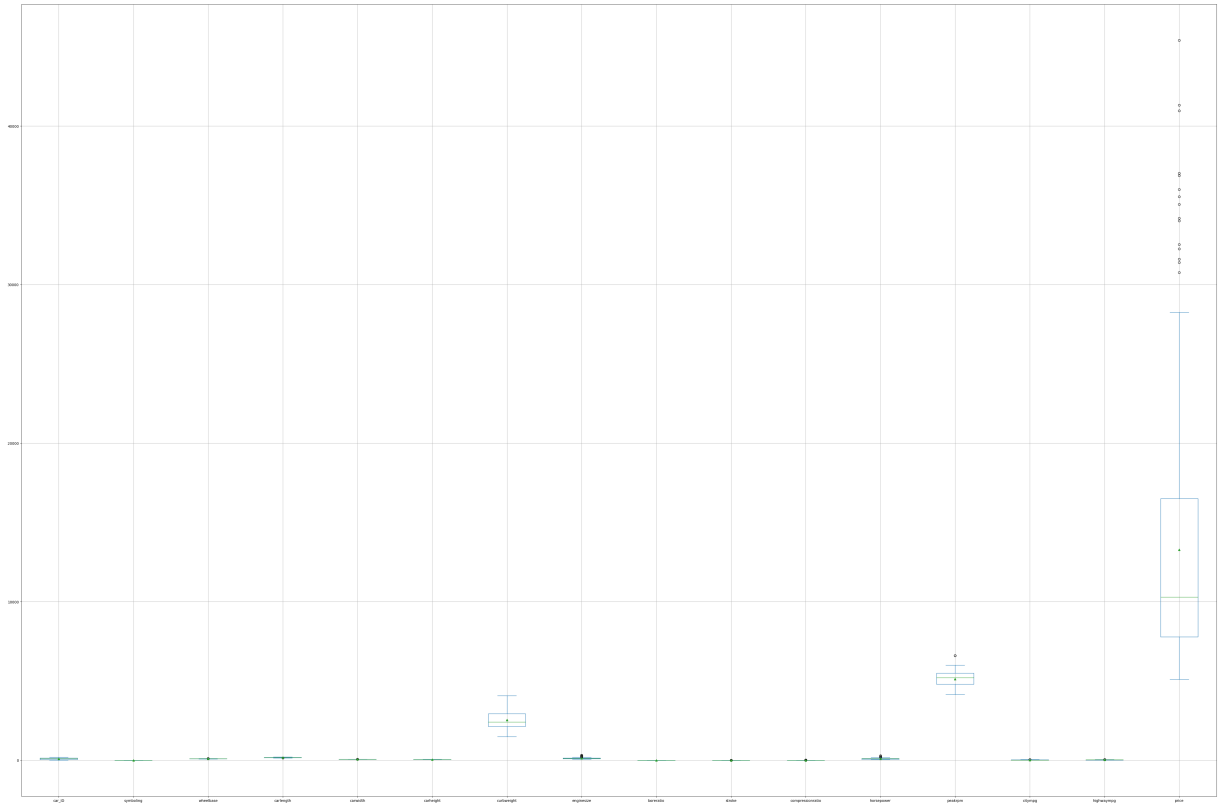
Finding outliers

For numeric values

```
df.describe()
```

	car_ID	symboling	wheelbase	carlength	carwidth	carheigh
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.00000
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.72487
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.44352
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.80000
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.00000
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.10000
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.50000
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.80000

```
#detect with box pllot
ax=df.plot.box(figsize=(60,40), showmeans=True)
ax.grid()
```



Handling Outliers

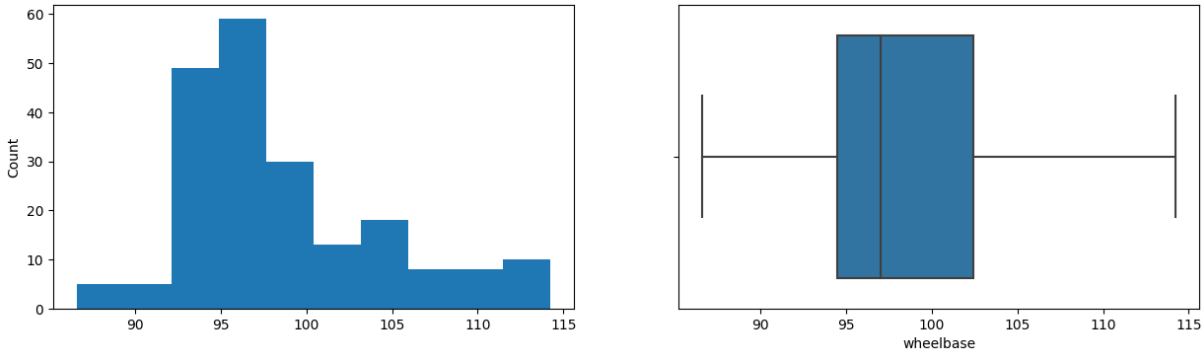
```
# Assuming you have a pandas DataFrame named df with the columns you mer
df1 = df.copy()  # Create a copy of the original DataFrame

columns_to_transform = ['wheelbase', 'carlength', 'carwidth', 'enginesi

# Define a function to perform the outlier transformation
def percentile_outlier_transformation(data, lower, upper):
    return np.where(data < lower, lower, np.where(data > upper, upper, d

for column_name in columns_to_transform:
    column_data = df1[column_name]

    # Calculate the 25th and 75th percentiles
    q25 = np.percentile(column_data, 25)
    q75 = np.percentile(column_data, 75)

    # Calculate the IQR (Interquartile Range)
    iqr = q75 - q25

    # Define the lower and upper bounds for outliers
    lower_bound = q25 - 1.5 * iqr
    upper_bound = q75 + 1.5 * iqr

    # Transform outliers using the defined function and update df1
    df1[column_name] = percentile_outlier_transformation(column_data, l
```

Showing box plot

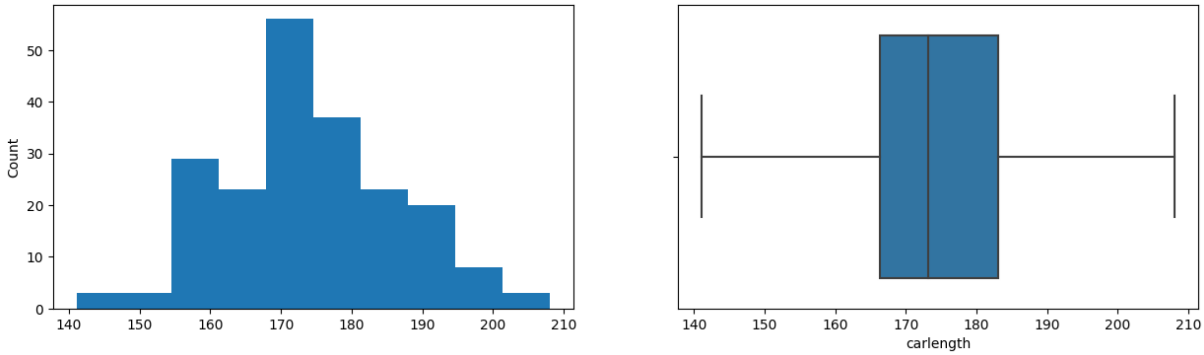
```
# Assuming you have already created the DataFrame df1

# List of numeric columns in df1
num_cols = ['wheelbase', 'carlength', 'carwidth', 'enginesize', 'stroke

# Iterate through each numeric column
for col in num_cols:
    print(col)
    print('Skew:', round(df1[col].skew(), 2))

    plt.figure(figsize=(15, 4))

    # Histogram
    plt.subplot(1, 2, 1)
    df1[col].hist(grid=False)
```

```
    plt.ylabel('Count')

    # Box Plot
    plt.subplot(1, 2, 2)
    sns.boxplot(x=df1[col])

    plt.show()
```

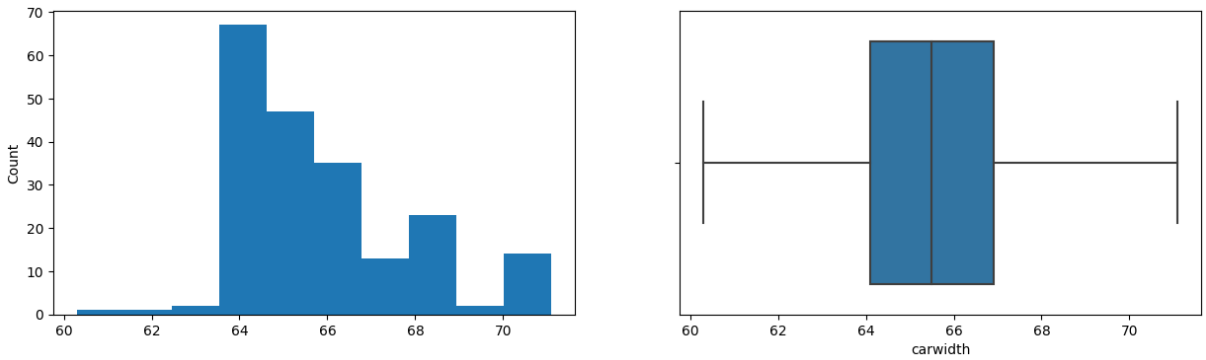
wheelbase
Skew: 0.92



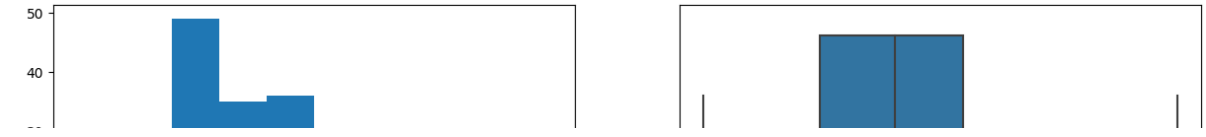
carlength
Skew: 0.16

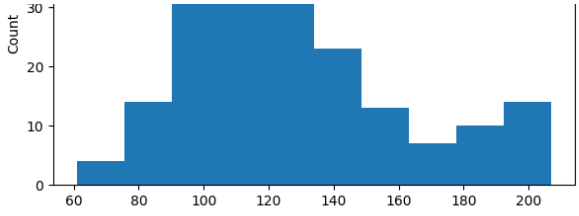


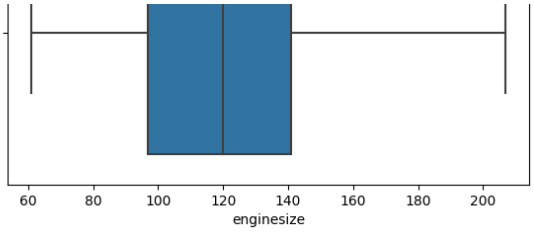
carwidth
Skew: 0.78



enginesize
Skew: 0.91

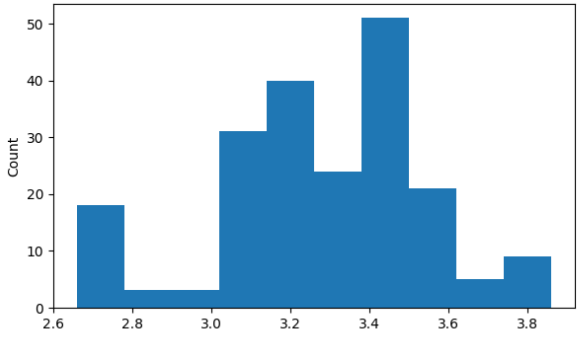


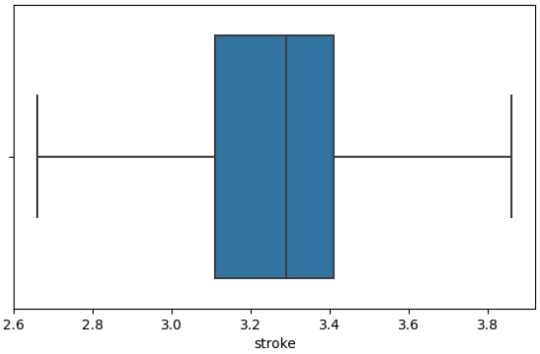




stroke

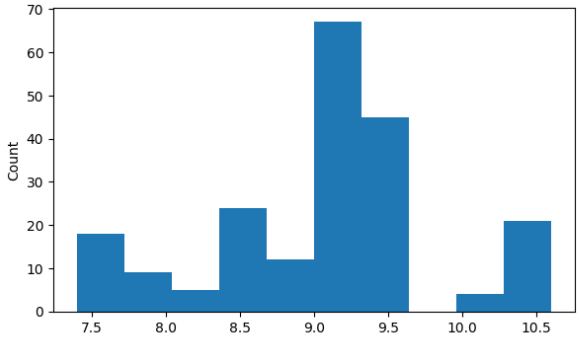
Skew: -0.38

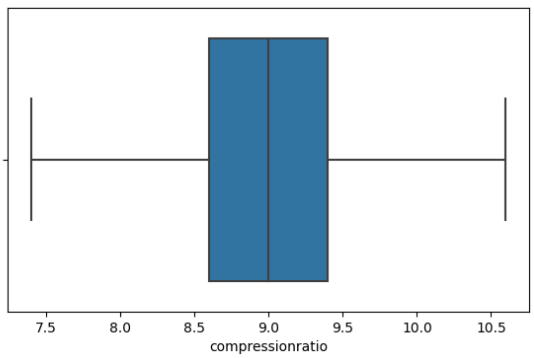




compressionratio

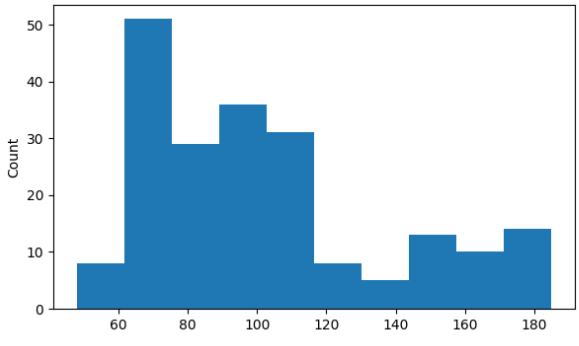
Skew: 0.04

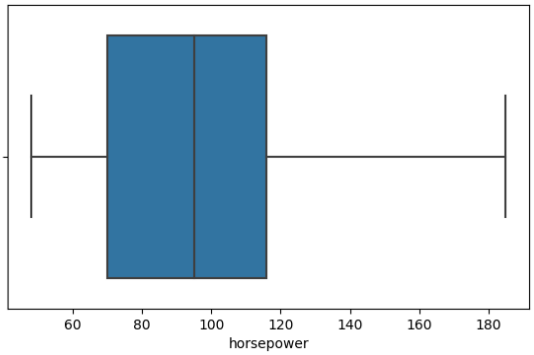




horsepower

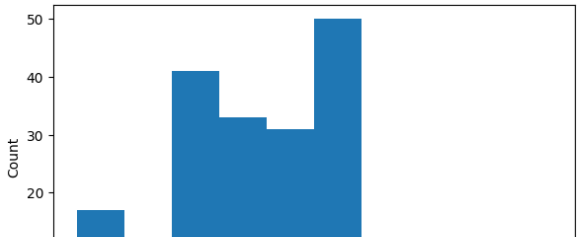
Skew: 0.81

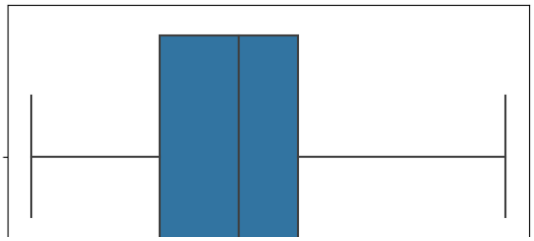


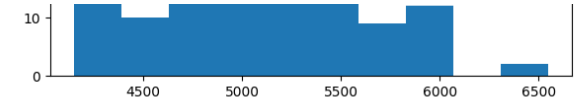


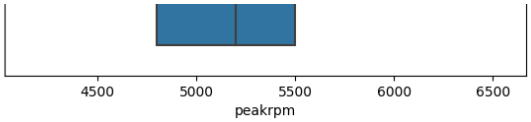
peakrpm

Skew: 0.05



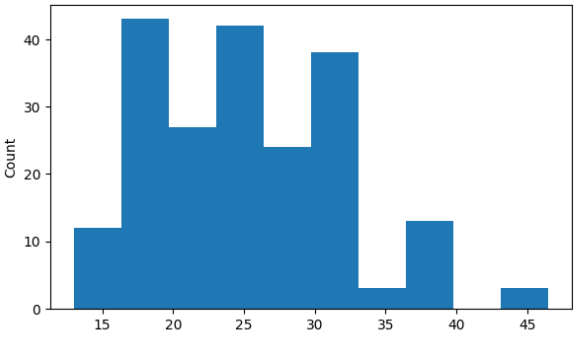


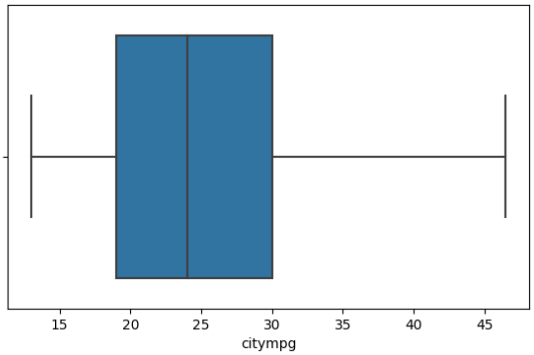




citympg

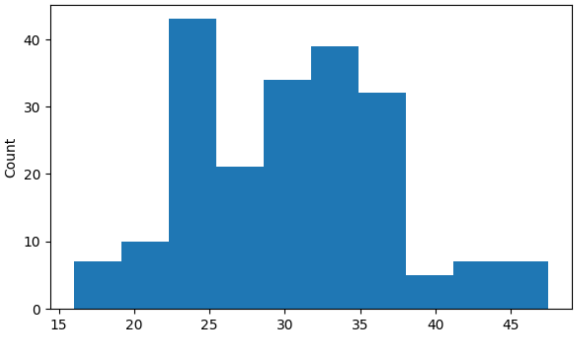
Skew: 0.6

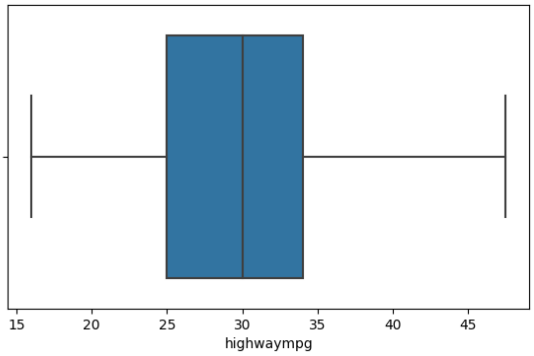




highwaympg

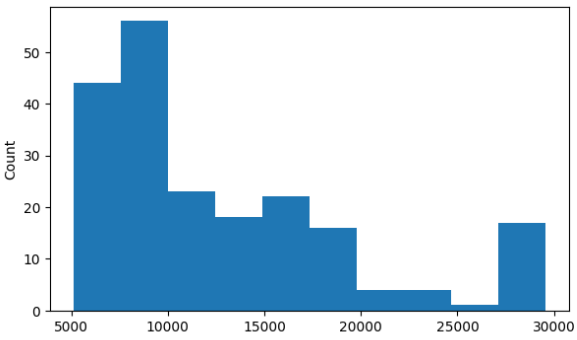
Skew: 0.35

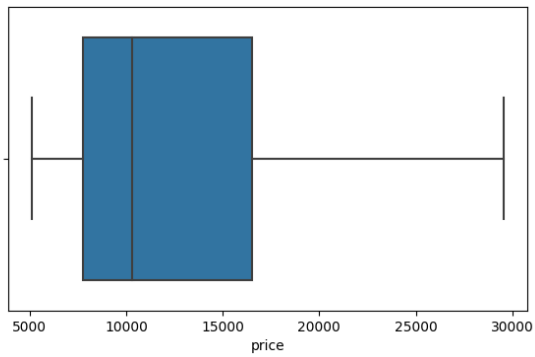




price

Skew: 1.22

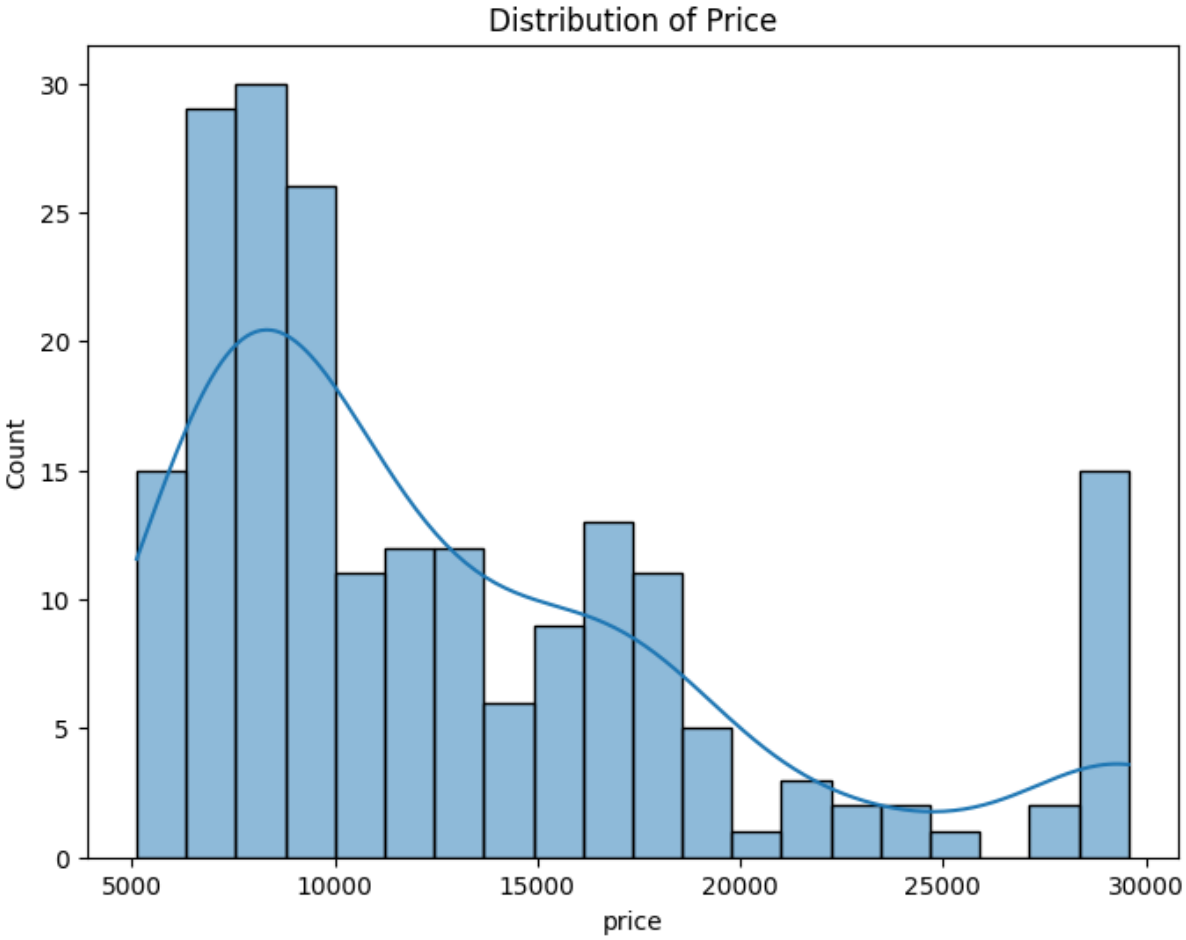




Bivariate Analysis

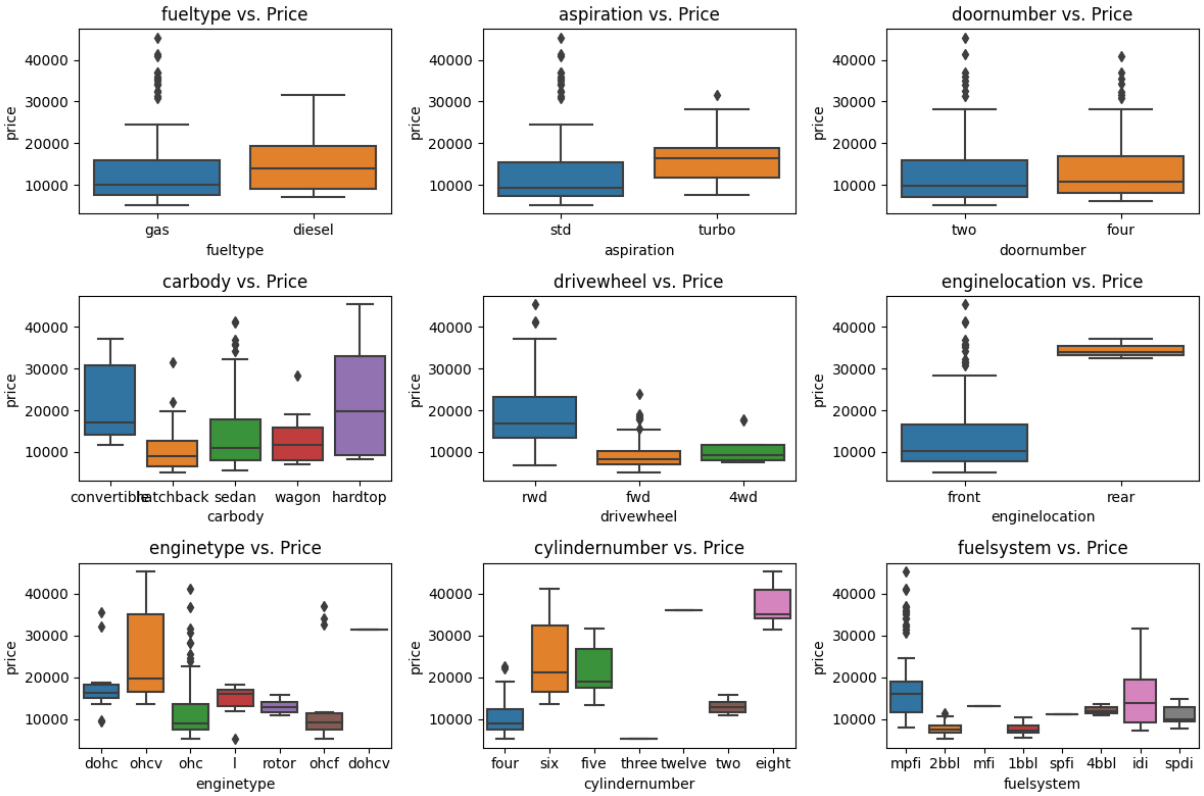
NUMERIC VARIABLES

```
# Price Analysis
plt.figure(figsize=(8, 6))
sns.histplot(data=df1['price'], bins=20, kde=True)
plt.title('Distribution of Price')
plt.show()
```



CATEGORICAL VARIABLE VS NUMERIC VARIABLE

```
# Categorical Feature vs. Price
plt.figure(figsize=(12, 8))
for feature in categorical_columns:
    plt.subplot(3, 3, categorical_columns.index(feature) + 1)
    sns.boxplot(data=df, x=feature, y='price')
    plt.title(f'{feature} vs. Price')
plt.tight_layout()
plt.show()
```



Multivariate Analysis

Correlation matrix

```
#dataframe correlation
df1.corr(method='spearman')
```

	car_ID	symboling	wheelbase	carlength	carwidth	c
car_ID	1.000000	-0.156530	0.196709	0.155379	0.148254	
symboling	-0.156530	1.000000	-0.538023	-0.396365	-0.253340	
wheelbase	0.196709	-0.538023	1.000000	0.912390	0.812479	
carlength	0.155379	-0.396365	0.912390	1.000000	0.888499	
carwidth	0.148254	-0.253340	0.812479	0.888499	1.000000	
carheight	0.263124	-0.523124	0.632940	0.525148	0.351021	
curbweight	0.124474	-0.256490	0.765289	0.890415	0.863463	
enginesize	0.088516	-0.176642	0.648119	0.782511	0.769726	
boreratio	0.272795	-0.169593	0.537006	0.638926	0.608874	
stroke	-0.160081	-0.016973	0.228100	0.188627	0.242237	
compressionratio	0.145062	0.019648	-0.122044	-0.187306	-0.141702	
horsepower	0.005364	-0.009407	0.504472	0.660144	0.688040	
peakrpm	-0.229643	0.282124	-0.311855	-0.269277	-0.198594	
citympg	0.055776	-0.018317	-0.492797	-0.670014	-0.687534	
highwaympg	0.021270	0.053254	-0.538578	-0.697897	-0.701063	
price	0.021091	-0.144992	0.681115	0.803628	0.810998	

```
#converting the categorical into numerical
df_num= df1
for col_name in df_num.columns:
    if(df_num[col_name ].dtype == 'object'):
     df_num[col_name] = df_num [col_name].astype('category')
     df_num[col_name] = df_num[ col_name].cat.codes
df_num
print(df_num)
```

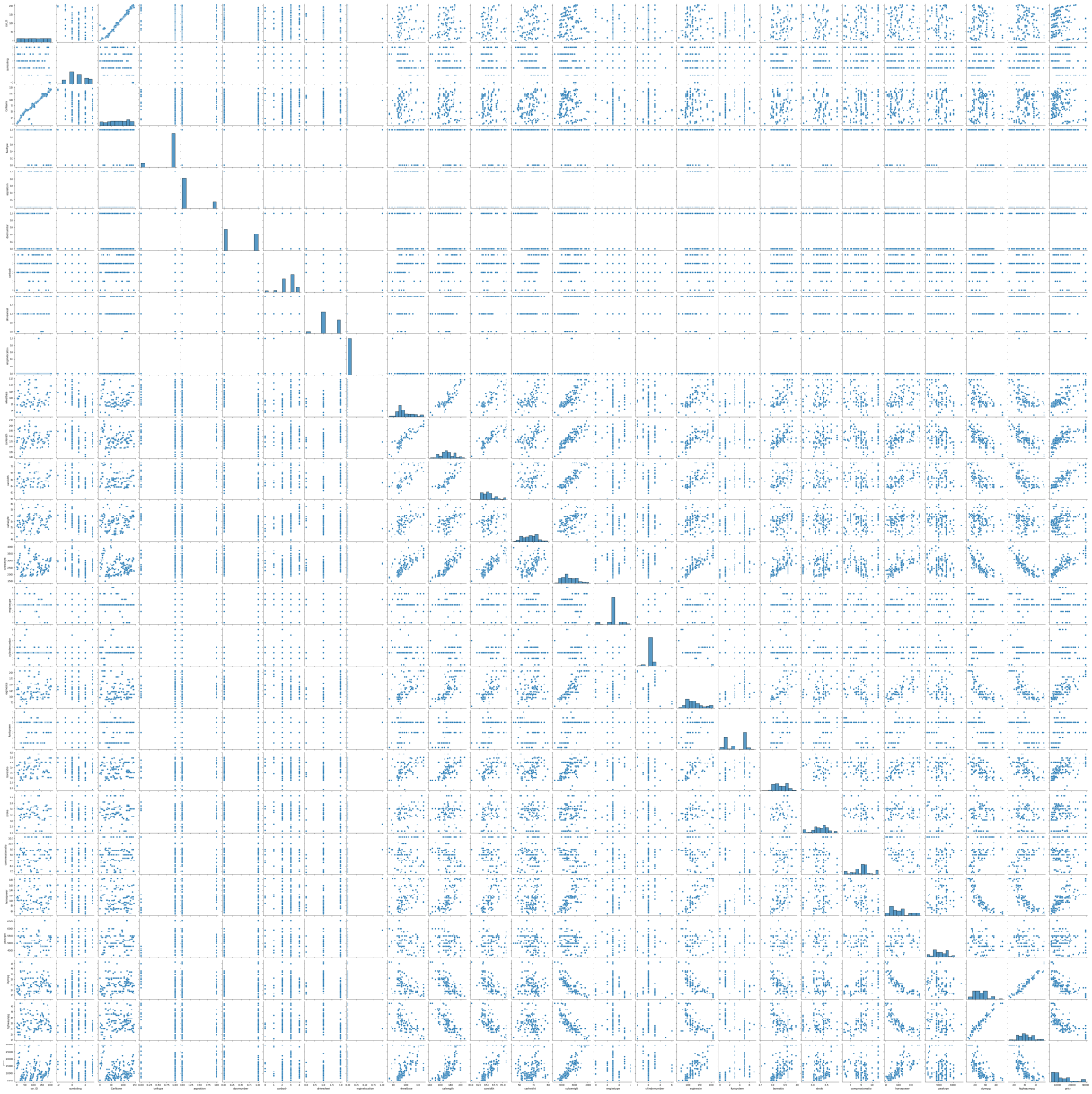
```
     car_ID  symboling  CarName  fueltype  aspiration  doornumber  c
0         1          3        2         1           0           1
1         2          3        3         1           0           1
2         3          1        1         1           0           1
3         4          2        4         1           0           0
```

```
4         5          2        5         1           0           0
..      ...        ...      ...       ...         ...         ...
200     201         -1      139         1           0           0
201     202         -1      138         1           1           0
202     203         -1      140         1           0           0
203     204         -1      142         0           1           0
204     205         -1      143         1           1           0

     drivewheel  enginelocation  wheelbase  ...  enginesize  fuelsys
0             2               0       88.6  ...       130.0
1             2               0       88.6  ...       130.0
2             2               0       94.5  ...       152.0
3             1               0       99.8  ...       109.0
4             0               0       99.4  ...       136.0
..          ...             ...        ...  ...         ...
200           2               0      109.1  ...       141.0
201           2               0      109.1  ...       141.0
202           2               0      109.1  ...       173.0
203           2               0      109.1  ...       145.0
204           2               0      109.1  ...       141.0

     boreratio  stroke  compressionratio  horsepower  peakrpm  citym
0         3.47    2.68               9.0       111.0   5000.0     21
1         3.47    2.68               9.0       111.0   5000.0     21
2         2.68    3.47               9.0       154.0   5000.0     19
3         3.19    3.40              10.0       102.0   5500.0     24
4         3.19    3.40               8.0       115.0   5500.0     18
..         ...     ...               ...         ...      ...      .
200       3.78    3.15               9.5       114.0   5400.0     23
201       3.78    3.15               8.7       160.0   5300.0     19
202       3.58    2.87               8.8       134.0   5500.0     18
203       3.01    3.40              10.6       106.0   4800.0     26
204       3.78    3.15               9.5       114.0   5400.0     19

     highwaympg    price
0          27.0  13495.0
1          27.0  16500.0
2          26.0  16500.0
3          30.0  13950.0
4          22.0  17450.0
..          ...      ...
200        28.0  16845.0
201        25.0  19045.0
202        23.0  21485.0
203        27.0  22470.0
204        25.0  22625.0

[205 rows x 26 columns]
```

Relationship between variables

```
sns.pairplot(df_num)
```

```
<seaborn.axisgrid.PairGrid at 0x7c550edb4e20>
```



Strength of correlation

```
df_num.corr()
```

	car_ID	symboling	CarName	fueltype	aspiration	d
car_ID	1.000000	-0.151621	0.967077	-0.125568	0.067729	
symboling	-0.151621	1.000000	-0.107095	0.194311	-0.059866	
CarName	0.967077	-0.107095	1.000000	-0.069435	0.019914	
fueltype	-0.125568	0.194311	-0.069435	1.000000	-0.401397	
aspiration	0.067729	-0.059866	0.019914	-0.401397	1.000000	
doornumber	-0.190352	0.664073	-0.171745	0.191491	-0.031792	
carbody	0.098303	-0.596135	0.099691	-0.147853	0.063028	
drivewheel	0.051406	-0.041671	-0.016129	-0.132257	0.066465	
enginelocation	0.051483	0.212471	0.055968	0.040070	-0.057191	
wheelbase	0.136702	-0.537515	0.025977	-0.314415	0.264436	
carlength	0.170636	-0.357612	0.053016	-0.212679	0.234539	
carwidth	0.059244	-0.235697	-0.066470	-0.239809	0.309321	
carheight	0.255960	-0.541038	0.201900	-0.284631	0.087311	
curbweight	0.071962	-0.227691	-0.049407	-0.217275	0.324902	
enginetype	-0.075130	0.050372	-0.090381	0.082695	-0.102963	
cylindernumber	-0.040912	0.197762	0.047154	0.110617	-0.133119	
enginesize	0.006263	-0.100869	-0.114889	-0.107971	0.164998	
fuelsystem	0.204898	0.091163	0.123845	0.041529	0.288086	
boreratio	0.260064	-0.130051	0.188598	-0.054451	0.212614	
stroke	-0.157469	0.006560	-0.182123	-0.266315	0.241716	
compressionratio	0.158596	-0.070206	0.147068	-0.648473	-0.140192	
horsepower	-0.017595	0.064578	-0.106263	0.170401	0.280095	
peakrpm	-0.205532	0.274328	-0.139731	0.478041	-0.183476	
citympg	0.018890	-0.038222	0.096412	-0.258696	-0.202875	

highwaympg	0.023471	0.027178	0.115906	-0.194567	-0.257215
price	-0.089603	-0.092705	-0.210533	-0.142586	0.238992

26 rows × 26 columns

```
df_num.corr(method='spearman')
```

	car_ID	symboling	CarName	fueltype	aspiration	d
car_ID	1.000000	-0.156530	0.973241	-0.125568	0.067729	
symboling	-0.156530	1.000000	-0.116181	0.209655	-0.076620	
CarName	0.973241	-0.116181	1.000000	-0.073206	0.019934	
fueltype	-0.125568	0.209655	-0.073206	1.000000	-0.401397	
aspiration	0.067729	-0.076620	0.019934	-0.401397	1.000000	
doornumber	-0.190352	0.679462	-0.176908	0.191491	-0.031792	
carbody	0.110600	-0.608863	0.101168	-0.170138	0.044720	
drivewheel	0.061609	-0.073682	0.002806	-0.131019	0.082006	
enginelocation	0.051483	0.189089	0.049770	0.040070	-0.057191	
wheelbase	0.196709	-0.538023	0.109008	-0.276784	0.226850	
carlength	0.155379	-0.396365	0.062974	-0.193855	0.246698	
carwidth	0.148254	-0.253340	0.053703	-0.232176	0.306708	
carheight	0.263124	-0.523124	0.214931	-0.298162	0.108209	
curbweight	0.124474	-0.256490	0.028087	-0.195995	0.342186	
enginetype	-0.017929	0.073868	-0.047261	0.165467	-0.145859	
cylindernumber	0.027526	0.096895	0.093981	0.148020	-0.172255	
enginesize	0.088516	-0.176642	-0.010173	-0.132261	0.231444	
fuelsystem	0.200356	0.084665	0.142218	0.074500	0.297505	
boreratio	0.272795	-0.169593	0.203184	-0.043695	0.192713	
stroke	-0.160081	-0.016973	-0.189605	-0.321903	0.234616	
compressionratio	0.145062	0.019648	0.143462	-0.515154	-0.135760	
horsepower	0.005364	-0.009407	-0.076935	0.159861	0.319171	

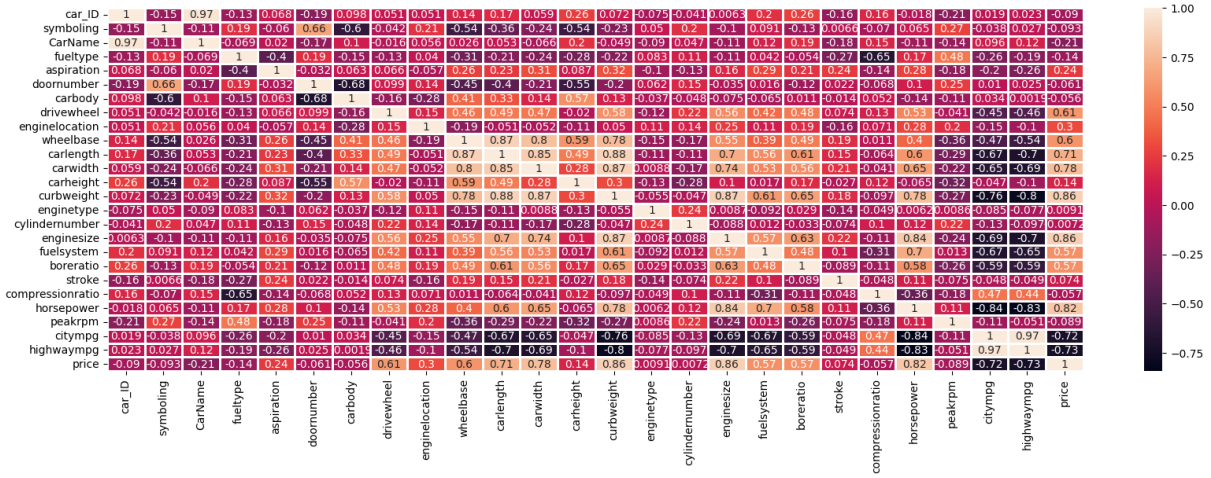
peakrpm	-0.229643	0.282124	-0.173220	0.454973	-0.131227
citympg	0.055776	-0.018317	0.123517	-0.234219	-0.218351
highwaympg	0.021270	0.053254	0.097978	-0.157978	-0.277593
price	0.021091	-0.144992	-0.067695	-0.141154	0.310093

26 rows × 26 columns

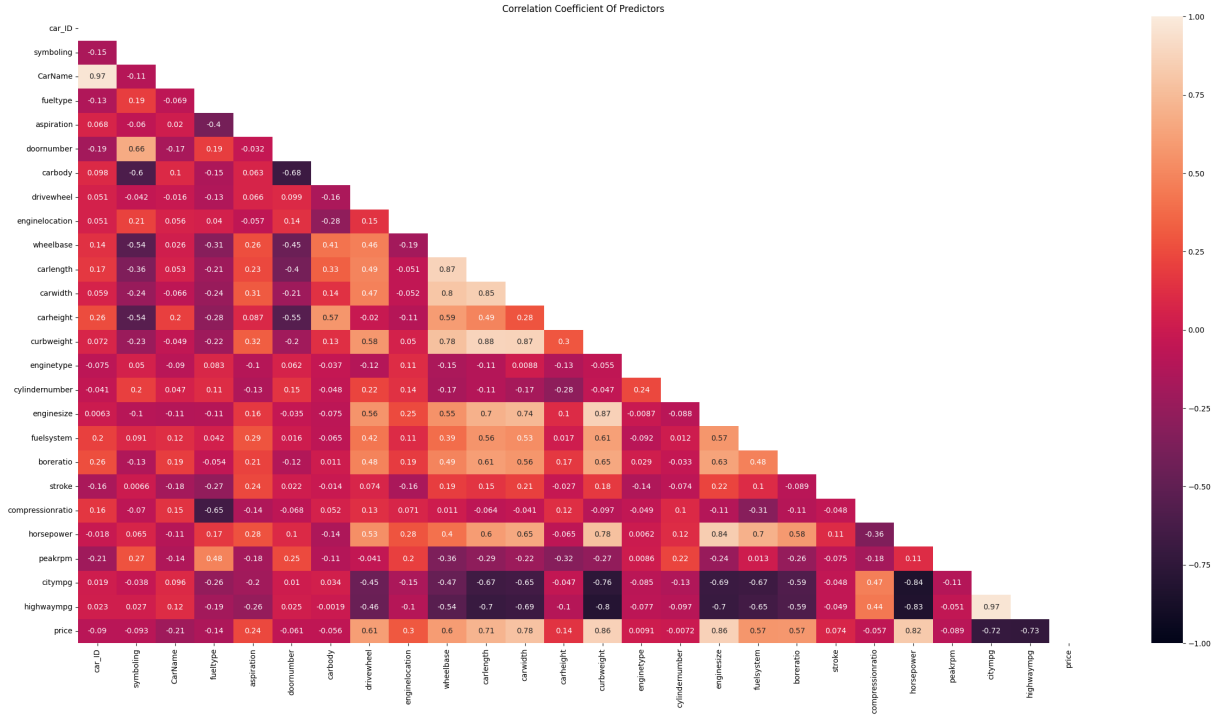
Strength of correlation by visualization

```
sns.heatmap(df_num.corr(),linewidths=1,annot=True)
```

```
<Axes: >
```



```
# set figure size
plt.figure(figsize=(30,15))

# Generate a mask to onlyshow the bottom triangle
mask = np.triu(np.ones_like(df_num.corr(), dtype=bool))

# generate heatmap
sns.heatmap(df_num.corr(), annot=True, mask=mask, vmin=-1, vmax=1)
plt.title('Correlation Coefficient Of Predictors')
plt.show()
```



Pairs of each variables

```
correlation_matrix=df_num.corr()
correlation_pairs=correlation_matrix.unstack()
correlation_pairs
```

```
car_ID  car_ID          1.000000
        symboling      -0.151621
        CarName         0.967077
        fueltype       -0.125568
        aspiration      0.067729
                          ...
price   horsepower      0.821715
        peakrpm        -0.088630
        citympg        -0.718290
        highwaympg     -0.733692
        price           1.000000
Length: 676, dtype: float64
```

Sorting pairs

```
sorted_pairs=correlation_pairs.sort_values()
sorted_pairs
```

```
horsepower        citympg            -0.842676
citympg           horsepower         -0.842676
horsepower        highwaympg         -0.830011
highwaympg        horsepower         -0.830011
                  curbweight         -0.803960
                                        ...
compressionratio  compressionratio    1.000000
horsepower        horsepower         1.000000
peakrpm           peakrpm            1.000000
carheight         carheight          1.000000
price             price              1.000000
Length: 676, dtype: float64
```

Grouped sorted pairs

```
corr_matrix = df_num.corr()

positive_pairs = []
negative_pairs = []
no_correlation = []

for i in range(len(corr_matrix.columns)):
    for j in range(i+1, len(corr_matrix.columns)):
```

```
        col1 = corr_matrix.columns[i]
        col2 = corr_matrix.columns[j]
        correlation = corr_matrix.loc[col1, col2]
        if correlation > 0:
            positive_pairs.append((col1, col2, correlation))
        elif correlation < 0:
            negative_pairs.append((col1, col2, correlation))
        else:
            no_correlation.append((col1, col2, correlation))

df_positive = pd.DataFrame(positive_pairs, columns=['Column 1', 'Column
df_negative = pd.DataFrame(negative_pairs, columns=['Column 1', 'Column
df_no_correlation = pd.DataFrame(no_correlation, columns=['Column 1', '

print("Positive_pairs:")
print(df_positive)

print("Negative pairs:")
print(df_negative)


print("No correlation:")
print(df_no_correlation)
```

```
Positive_pairs:
            Column 1        Column 2  Correlation
0             car_ID         CarName     0.967077
1             car_ID      aspiration     0.067729
2             car_ID         carbody     0.098303
3             car_ID      drivewheel     0.051406
4             car_ID  enginelocation     0.051483
..               ...             ...          ...
163  compressionratio         citympg     0.468369
164  compressionratio      highwaympg     0.435872
165        horsepower         peakrpm     0.114871
166        horsepower           price     0.821715
167           citympg      highwaympg     0.966524

[168 rows x 3 columns]
Negative pairs:
       Column 1        Column 2  Correlation
0        car_ID       symboling    -0.151621
1        car_ID        fueltype    -0.125568
2        car_ID      doornumber    -0.190352
3        car_ID      enginetype    -0.075130
4        car_ID  cylindernumber    -0.040912
..          ...             ...          ...
152     peakrpm         citympg    -0.113602
153     peakrpm      highwaympg    -0.051123
154     peakrpm           price    -0.088630
155     citympg           price    -0.718290
156  highwaympg           price    -0.733692

[157 rows x 3 columns]
No correlation:
Empty DataFrame
Columns: [Column 1, Column 2, Correlation]
Index: []
```

```
df_positive
```

	Column 1	Column 2	Correlation
0	car_ID	CarName	0.967077
1	car_ID	aspiration	0.067729
2	car_ID	carbody	0.098303
3	car_ID	drivewheel	0.051406
4	car_ID	enginelocation	0.051483
...	...	...	...
163	compressionratio	citympg	0.468369
164	compressionratio	highwaympg	0.435872
165	horsepower	peakrpm	0.114871
166	horsepower	price	0.821715
167	citympg	highwaympg	0.966524

168 rows × 3 columns

```
df_negative
```

	Column 1	Column 2	Correlation
0	car_ID	symboling	-0.151621
1	car_ID	fueltype	-0.125568
2	car_ID	doornumber	-0.190352
3	car_ID	enginetype	-0.075130
4	car_ID	cylindernumber	-0.040912
...	...	...	...
152	peakrpm	citympg	-0.113602
153	peakrpm	highwaympg	-0.051123
154	peakrpm	price	-0.088630
155	citympg	price	-0.718290
156	highwaympg	price	-0.733692

157 rows × 3 columns

```
df_no_correlation
```

	Column 1	Column 2	Correlation

Sorted high pairs

```
high_corr=sorted_pairs[(sorted_pairs)>0.5]
high_corr
```

```
carwidth          fuelsystem          0.528580
fuelsystem        carwidth            0.528580
drivewheel        horsepower          0.532969
horsepower        drivewheel          0.532969
enginesize        wheelbase           0.554028
                                        ...
compressionratio  compressionratio    1.000000
horsepower        horsepower          1.000000
peakrpm           peakrpm             1.000000
carheight         carheight           1.000000
price             price               1.000000
Length: 108, dtype: float64
```

Target Variable analysis

Target Variable: Price

Correlation Values

```
df_num.corr()[['price']].sort_values('price', ascending=False)
```

	price
price	1.000000
curbweight	0.864597
enginesize	0.860063
horsepower	0.821715
carwidth	0.783230
carlength	0.712455
drivewheel	0.606814
wheelbase	0.595909
boreratio	0.572685
fuelsystem	0.570273
enginelocation	0.304551
aspiration	0.238992
carheight	0.142033
stroke	0.073830
enginetype	0.009102
cylindernumber	-0.007226
carbody	-0.055794
compressionratio	-0.056573
doornumber	-0.061434
peakrpm	-0.088630
car_ID	-0.089603
symboling	-0.092705
fueltype	-0.142586
CarName	-0.210533
citympg	-0.718290
highwaympg	-0.733692

Relationship between variables

```
sns.pairplot(df_num,hue="price", palette="husl")
```

Strength of correlation

```
sns.heatmap(df_num.corr()[['price']].sort_values(by='price',ascending=Fa
```

Direction of correlation

```
# Create a scatter plot matrix
sns.pairplot(df_num, x_vars=['price'],
             y_vars=['curbweight', 'enginesize', 'horsepower'], kind='s

# Show plot
sns.show()
```

Grouping correlation

```
# Calculate the correlation matrix
corr_matrix = df_num.corr()

# Get the correlation coefficients of 'TAX TO GOVERNMENT' with other va
tax_corr = corr_matrix['price'].drop('price')

# Define the correlation categories
categories = {
    'Very Strong Positive Correlation': (0.80, 1),
    'Strong Positive Correlation': (0.60, 0.79),
    'Moderate Positive Correlation': (0.40, 0.59),
    'Weak Positive Correlation': (0.20, 0.39),
    'Very Weak Positive Correlation': (0.00, 0.19),
    'Very Strong Negative Correlation': (-1.00, -0.80),
    'Strong Negative Correlation': (-0.79, -0.60),
    'Moderate Negative Correlation': (-0.59, -0.40),
    'Weak Negative Correlation': (-0.39, -0.20),
    'Very Weak Negative Correlation': (-0.19, 0.01)
}

# Create an empty dictionary to hold the dataframes
df_dict = {}

# Categorize the correlation coefficients and create a separate datafra
for category, (min_value, max_value) in categories.items():
    temp_df = tax_corr[(tax_corr >= min_value) & (tax_corr <= max_value
    df_dict[category] = temp_df

# Print each dataframe
for category, df in df_dict.items():
    print(f"{category}:\n{df}\n")
```

Observations:

1)Univariate analysis:

Found outliers in numerical values.

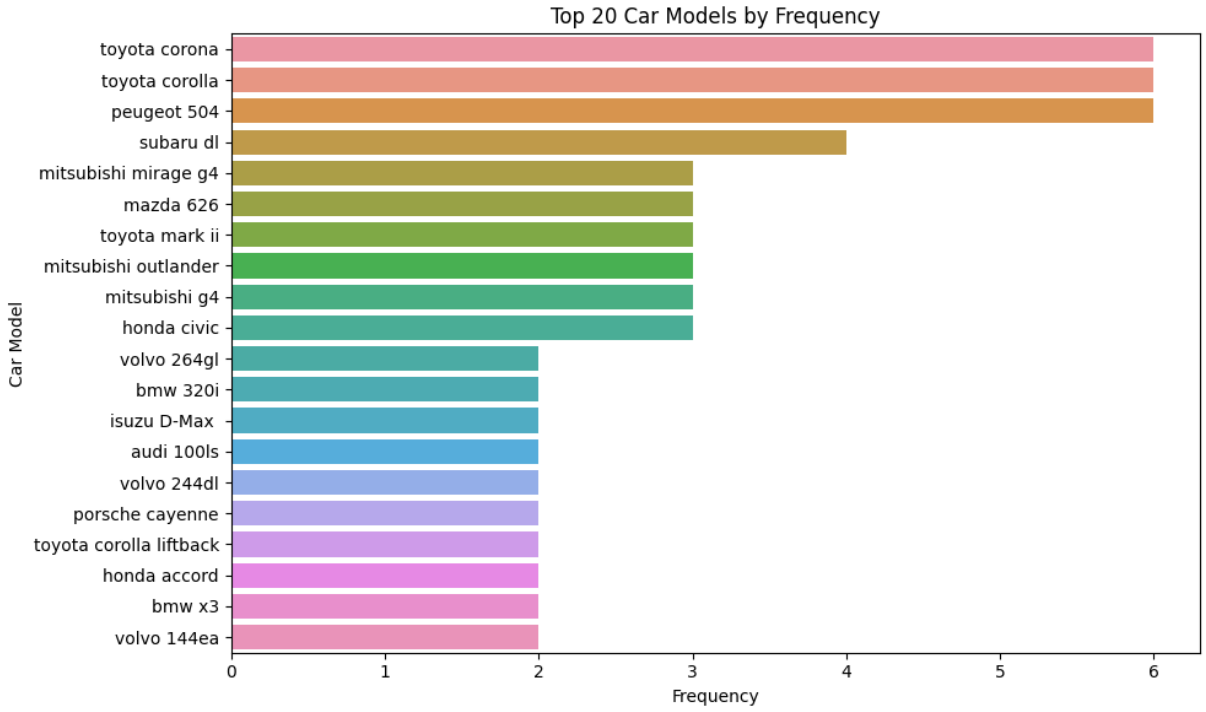
Handled outliers by percentile method.

2)Bivariate analysis: Price distribution shows right skew.

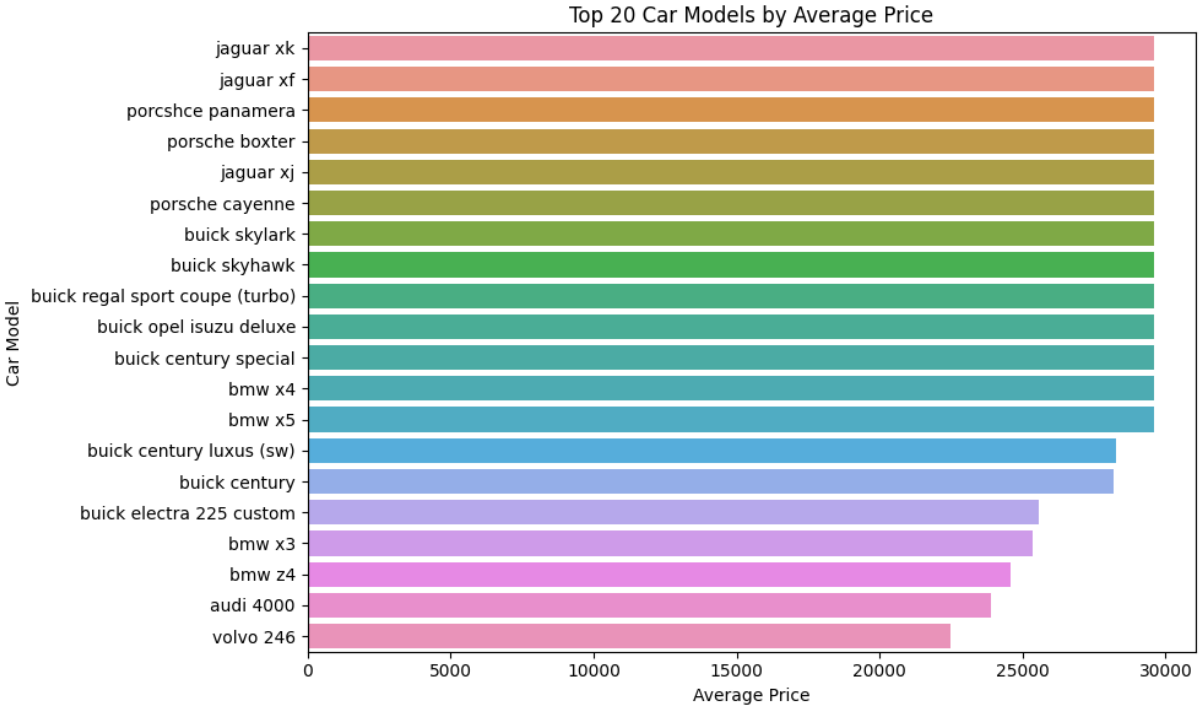
3**)Multivariate analysis:**Very Strong Positive Correlation: price curbweight 0.864597 enginesize 0.860063 horsepower 0.821715

4-ANALYSING WITH QUESTIONS

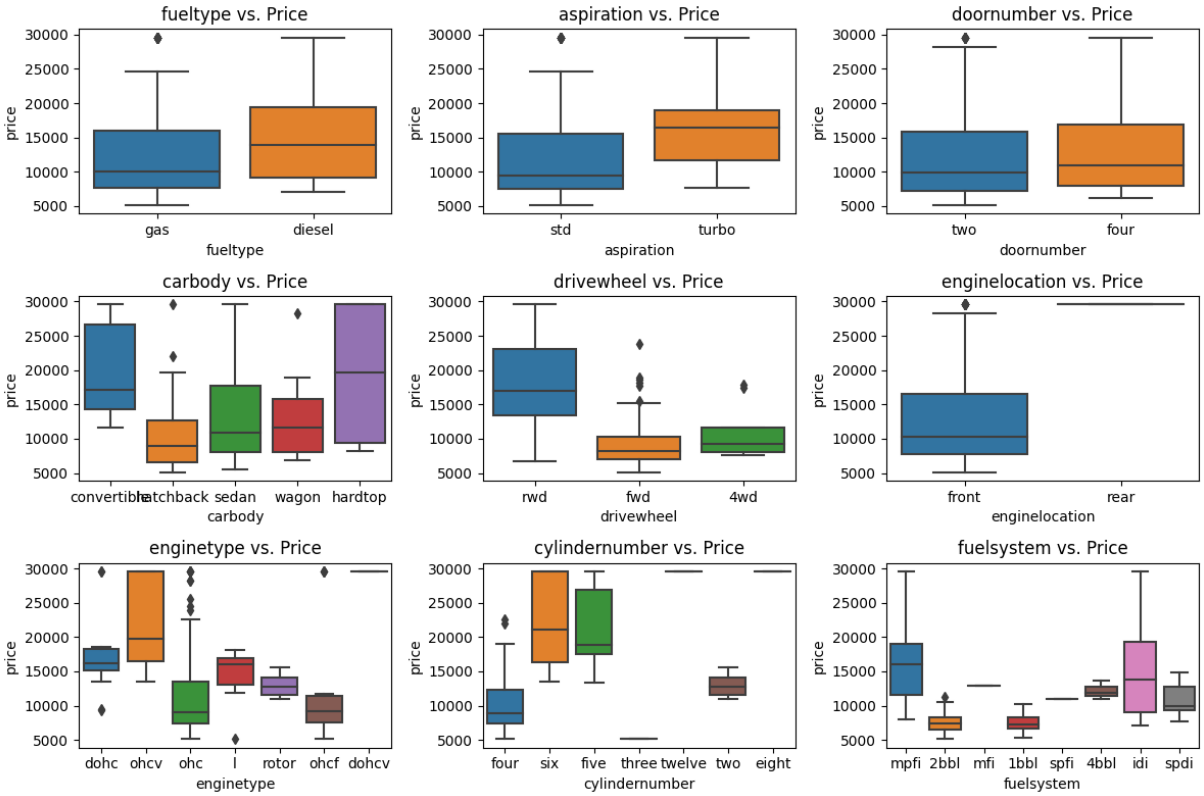
```
n = 20  # Number of top car models to plot
top_car_models = df1['CarName'].value_counts().head(n)

plt.figure(figsize=(10, 6))
sns.barplot(x=top_car_models.values, y=top_car_models.index)
plt.title(f'Top {n} Car Models by Frequency')
plt.xlabel('Frequency')
plt.ylabel('Car Model')
plt.tight_layout()
plt.show()
```



```
# Calculate average price for each car model
avg_prices_by_car = df1.groupby('CarName')['price'].mean().sort_values(a

# Plot top N car models by average price
n = 20  # Number of top car models to plot
top_car_models = avg_prices_by_car.head(n)

plt.figure(figsize=(10, 6))
sns.barplot(x=top_car_models.values, y=top_car_models.index)
plt.title(f'Top {n} Car Models by Average Price')
plt.xlabel('Average Price')
plt.ylabel('Car Model')
plt.tight_layout()
plt.show()
```

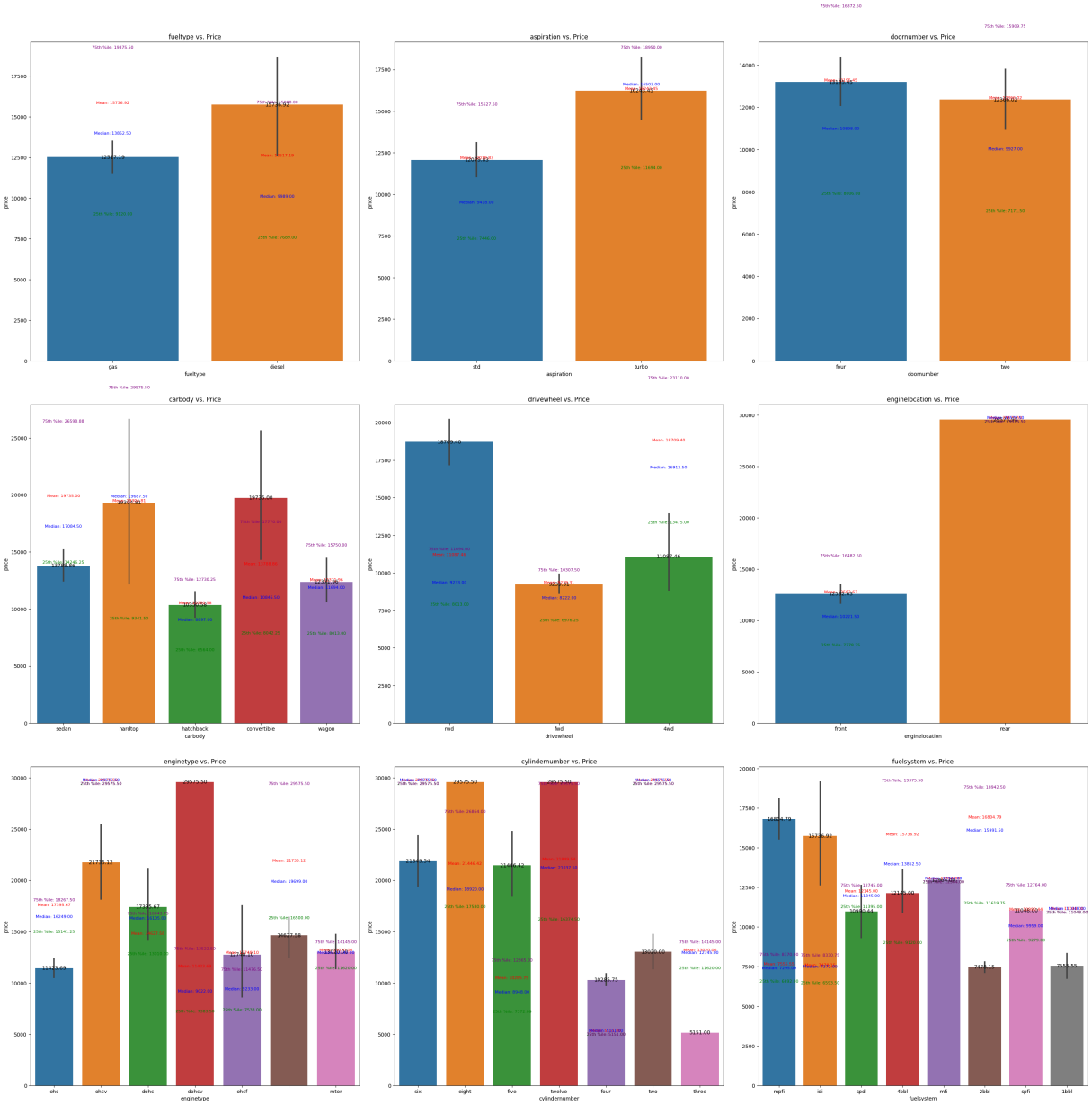


```
# Categorical Feature vs. Price
plt.figure(figsize=(12, 8))
for feature in categorical_columns:
    plt.subplot(3, 3, categorical_columns.index(feature) + 1)
    sns.boxplot(data=df1, x=feature, y='price')
    plt.title(f'{feature} vs. Price')
plt.tight_layout()
plt.show()
```




```
# Set a larger figure size
plt.figure(figsize=(30, 30))

# Sort the DataFrame by 'price' in descending order
df_sorted = df1.sort_values(by='price', ascending=False)

# Create a bar plot for each categorical feature
for i, feature in enumerate(categorical_columns, 1):
```

```
    plt.subplot(3, 3, i)
    ax = sns.barplot(data=df_sorted, x=feature, y='price')
    plt.title(f'{feature} vs. Price')

    # Annotate each bar with its value
    for p in ax.patches:
        ax.annotate(f'{p.get_height():.2f}', (p.get_x() + p.get_width()

    # Calculate and display statistics
    mean_price = df_sorted.groupby(feature)['price'].mean()
    median_price = df_sorted.groupby(feature)['price'].median()
    percentile_25 = df_sorted.groupby(feature)['price'].quantile(0.25)
    percentile_75 = df_sorted.groupby(feature)['price'].quantile(0.75)

    for j, (mean, median, p25, p75) in enumerate(zip(mean_price, median_
        ax.text(j, mean, f'Mean: {mean:.2f}', ha='center', va='bottom',
        ax.text(j, median, f'Median: {median:.2f}', ha='center', va='bo
        ax.text(j, p25, f'25th %ile: {p25:.2f}', ha='center', va='top',
        ax.text(j, p75, f'75th %ile: {p75:.2f}', ha='center', va='top',

plt.tight_layout()
plt.show()
```




```
# Perform ANOVA for each categorical feature
results = []
for feature in categorical_columns:
```

```
    model = ols(f'price ~ {feature}', data=df1).fit()
    anova_table = sm.stats.anova_lm(model, typ=2)
    results.append((feature, anova_table['PR(>F)'][0]))

# Create a DataFrame with ANOVA results
anova_results = pd.DataFrame(results, columns=['Categorical Feature', '

# Calculate summary statistics for each category
summary_stats = df1.groupby(categorical_columns)['price'].describe()

# Display ANOVA results
print("ANOVA results:")
print(anova_results)

# Display summary statistics
print("\nSummary statistics:")
print(summary_stats)
```

```
                                                ohcv       six
          sedan     fwd        front           ohc        four


    ornumber carbody   drivewheel enginelocation enginetype cylindernumb
    ur       hatchback fwd        front           ohc        four
             sedan     fwd        front           ohc        four
                       rwd        front           ohc        four
    o        sedan     fwd        front           ohc        four
    ur       sedan     fwd        front           ohc        four

    o        hatchback fwd        front           ohc        four
                       rwd        front           ohc        four

                                                ohcv       six
             sedan     fwd        front           ohc        four


    ornumber carbody   drivewheel enginelocation enginetype cylindernumb
    ur       hatchback fwd        front           ohc        four
             sedan     fwd        front           ohc        four
                       rwd        front           ohc        four
    o        sedan     fwd        front           ohc        four
    ur       sedan     fwd        front           ohc        four

    o        hatchback fwd        front           ohc        four
                       rwd        front           ohc        four

                                                ohcv       six
             sedan     fwd        front           ohc        four
```

```
ornumber carbody   drivewheel enginelocation enginetype cylindernumb
ur       hatchback fwd        front          ohc        four
         sedan     fwd        front          ohc        four
                   rwd        front          ohc        four
o        sedan     fwd        front          ohc        four
ur       sedan     fwd        front          ohc        four

o        hatchback fwd        front          ohc        four
                   rwd        front          ohc        four

                                             ohcv       six
         sedan     fwd        front          ohc        four


ornumber carbody   drivewheel enginelocation enginetype cylindernumb
ur       hatchback fwd        front          ohc        four
         sedan     fwd        front          ohc        four
                   rwd        front          ohc        four
o        sedan     fwd        front          ohc        four
ur       sedan     fwd        front          ohc        four

o        hatchback fwd        front          ohc        four
                   rwd        front          ohc        four

                                             ohcv       six
         sedan     fwd        front          ohc        four
```

Observations:

- Most top car is Toyota corona by frequency.
- Most top car is Jaguar xk with average price.
- Here are insights about each of the specified categorical features vs. price along with the provided metrics:

1. **Fuel Type vs. Price**:
    - Diesel vs. Gas: Diesel cars tend to have lower prices compared to gas cars. The mean price for diesel is generally lower, but there is no significant difference in variability between the two fuel types.
2. **Aspiration Type vs. Price**:
    - Std vs. Turbo: Cars with turbochargers (turbo) have higher average prices

compared to standard aspiration (std) cars. Turbocharged cars also exhibit a wider price range.

3. **Number of Doors vs. Price**:
    - Four vs. Two: Cars with four doors generally have more consistent prices than those with two doors. Two-door cars have a wider price distribution.
4. **Car Body Type vs. Price**:
    - Hatchback vs. Sedan: Hatchbacks typically have lower prices compared to sedans. The mean price for hatchbacks is lower, with less variability.
5. **Drive Wheel Type vs. Price**:
    - FWD vs. RWD: Cars with front-wheel drive (FWD) generally have lower average prices compared to rear-wheel drive (RWD) cars. RWD cars also show greater price variation.
6. **Engine Location vs. Price**:
    - Front Engine: Regardless of engine location, cars with the engine in the front tend to have consistent pricing.
7. **Engine Type vs. Price**:
    - OHC vs. OHCV: Cars with overhead camshaft (OHC) engines have more consistent prices compared to those with overhead camshaft, V-shaped (OHCV) engines.
8. **Number of Cylinders vs. Price**:
    - Four Cylinders vs. Six Cylinders: Cars with four cylinders have more consistent prices compared to six-cylinder cars, with OHCV engines generally having higher prices.
9. **Fuel System vs. Price**:
    - IDI vs. MPFI/SPDI: Cars with an indirect diesel injection (IDI) system have lower average prices, with relatively consistent pricing. Cars with multi-point fuel injection (MPFI) and single-point fuel injection (SPDI) systems exhibit more varied pricing.

5-DATA PRE-PROCESSING FOR ML

1) Data Transformation

Define categorical and numerical columns

```
# Extract brand and model from CarName
df1['brand'] = df1['CarName'].apply(lambda x: x.split(' ')[0])
df1['model'] = df1['CarName'].apply(lambda x: ' '.join(x.split(' ')[1:]
```

```
# Define categorical and numerical columns
categorical_columns = ['fueltype', 'aspiration', 'doornumber', 'carbody
                       'enginelocation', 'enginetype', 'cylindernumber'
numerical_columns = ['wheelbase', 'carlength', 'carwidth', 'carheight',
                     'enginesize', 'boreratio', 'stroke', 'compressionra
                     'peakrpm', 'citympg', 'highwaympg']
```

Encoding Categorical Variables

```
label_encoder = LabelEncoder()
for column in categorical_columns:
    df1[column] = label_encoder.fit_transform(df1[column])
```

Feature engineering

```
df1['power_to_weight_ratio'] = df1['horsepower'] / df1['curbweight']
for column in numerical_columns:
    df1[f'{column}_squared'] = df1[column] ** 2
df1['log_enginesize'] = np.log(df1['enginesize'] + 1)
```

Feature Scaling

```
# Feature scaling
scaler = StandardScaler()
df1[numerical_columns] = scaler.fit_transform(df1[numerical_columns])
```

2) Data Reduction

```
# Data Reduction (PCA)
pca = PCA(n_components=10)  # Adjust the number of components as needed
df_pca = pca.fit_transform(df1[numerical_columns])
df_pca = pd.DataFrame(df_pca, columns=[f'PCA_{i}' for i in range(1, 11)
df1 = pd.concat([df1, df_pca], axis=1)
```

3) Feature Selection

```
# 6. Feature Selection
X = df1.drop(['price', 'CarName'], axis=1)
y = df1['price']
```

6-BUILDING ML MODEL

Data Splitting

```
# Splitting the dataset
X = df1.drop(['price', 'CarName'], axis=1)  # Include the engineered fe
y = df1['price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2
```

Model Selection

```
# Create a Linear Regression model
model = LinearRegression()
```

Model Training

```
# Model training
model = LinearRegression()
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)
```

Model Evaluation

```
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2_square = r2_score(y_test,y_pred)
print(f" R-squared: {r2_square}")
print(f'Mean Squared Error: {mse}')
```

```
 R-squared: 0.8891064927306754
Mean Squared Error: 5873117.457896145
```

```
pred_df=pd.DataFrame({'Actual Value':y_test,'Predicted Value':y_pred,'D
pred_df
```

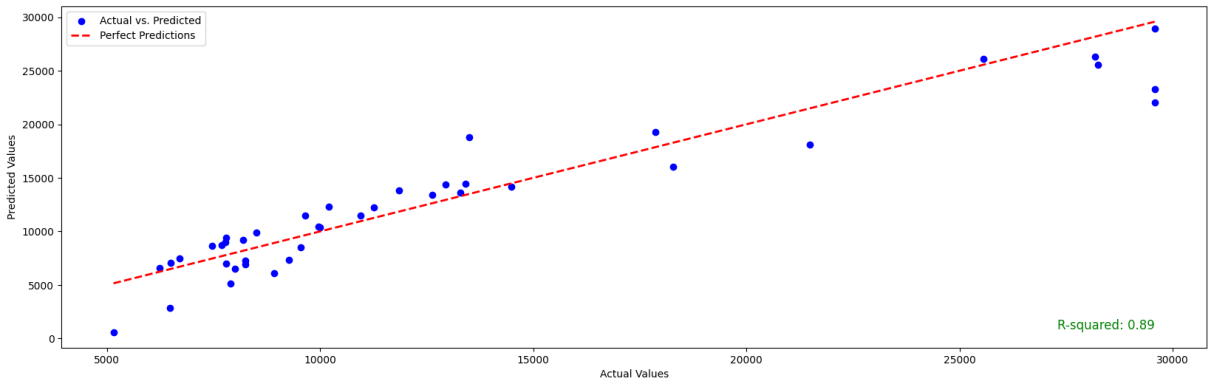
	Actual Value	Predicted Value	Difference
15	29575.500	22063.504922	7511.995078
9	17859.167	19267.605962	-1408.438962
100	9549.000	8503.391250	1045.608750
132	11850.000	13812.316311	-1962.316311
68	28248.000	25553.648429	2694.351571
95	7799.000	7005.765450	793.234550
159	7788.000	9435.511546	-1647.511546
162	9258.000	7346.711123	1911.288877
147	10198.000	12297.799095	-2099.799095
182	7775.000	9012.877430	-1237.877430
191	13295.000	13600.806435	-305.806435

164	8238.000	7279.304481	958.695519
65	18280.000	16049.657255	2230.342745
175	9988.000	10393.174297	-405.174297
73	29575.500	28907.198059	668.301941
152	6488.000	7031.650590	-543.650590
18	5151.000	596.705802	4554.294198
82	12629.000	13389.284838	-760.284838
86	8189.000	9182.373248	-993.373248
143	9960.000	10467.035330	-507.035330
60	8495.000	9865.028981	-1370.028981
101	13499.000	18762.220741	-5263.220741
98	8249.000	6954.273291	1294.726709
30	6479.000	2866.294149	3612.705851
25	6692.000	7466.243034	-774.243034
16	29575.500	23311.260234	6264.239766
168	9639.000	11505.532461	-1866.532461
195	13415.000	14424.001807	-1009.001807
97	7999.000	6503.473671	1495.526329
194	12940.000	14407.851726	-1467.851726
67	25552.000	26106.406607	-554.406607
120	6229.000	6564.382328	-335.382328
154	7898.000	5096.963617	2801.036383
202	21485.000	18129.832968	3355.167032
79	7689.000	8720.599540	-1031.599540
69	28176.000	26316.815497	1859.184503
145	11259.000	12262.024616	-1003.024616
55	10945.000	11478.973112	-533.973112
45	8916.500	6097.238232	2819.261768

84	14489.000	14172.604259	316.395741
146	7463.000	8630.349565	-1167.349565

Model Visualization

```
# Create a scatter plot of the actual data points
plt.scatter(y_test, y_pred, color='blue', label='Actual vs. Predicted')

# Add a diagonal line representing perfect predictions
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], co

# Label the axes
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')

# Calculate R-squared
r2_square = r2_score(y_test, y_pred)

# Add R-squared score as text on the plot (bottom-right corner)
plt.text(y_test.max(), y_pred.min(), f'R-squared: {r2_square:.2f}', hor

# Add a legend
plt.legend()

# Show the plot
plt.show()
```



Model Interpretation

```
# For Linear Regression, you can inspect coefficients
coefficients = model.coef_
intercept = model.intercept_

print("Coefficients:", coefficients)
print("Intercept:", intercept)
```

```
Coefficients: [-1.50437753e+01 -1.46560103e+01 -8.22917466e+02  1.85
 -3.10731480e+02 -3.66935102e+02  7.97011216e+02  1.37737716e+04
  2.40346240e+03  2.25413456e+02 -1.05193192e+04  8.68052551e+03
 -2.90168306e+03  4.72689411e+01 -3.27384021e+02 -3.01715351e+04
  2.43823490e+02 -1.23390928e+03  6.38182030e+02  5.52210587e+03
  2.51848334e+04 -6.86964111e+03 -3.03994118e+03  3.47927757e+03
  3.52677154e+00  1.18227864e+01 -8.91786749e+05 -1.32015360e+00
 -9.59926295e-01  8.13740050e+01 -5.80216530e+01  8.35623295e-04
  2.06111332e+00  1.15359329e+03  1.82614237e+02 -4.22133034e+02
 -1.01218346e+00  1.21722893e-03  7.11832937e+00  2.11344910e+00
  6.52026663e+04 -5.27403412e+03  2.45688840e+03  5.48380079e+03
 -7.25465456e+03  6.41008986e+02 -3.81668627e+03  2.58959292e+03
  9.03064907e+03 -5.60366383e+03 -1.96148495e+03]
Intercept: -454968.57619994384
```

Observations:

- Linear Regression Model Result:88.9

1. **R-squared (R^2):**
    - **High Explained Variance**: An R-squared value of 0.8891 means that the linear regression model explains approximately 88.91% of the variance in the 'price' of cars. This is a strong performance, indicating that a significant portion of the variability in car prices can be accounted for by the features used in the model.
    - **Model Fit**: The model fits the data well, capturing the relationship between the predictor variables (features) and the target variable (car prices).
    - **Predictive Power**: The higher the R-squared value, the better the model is

at making predictions based on the available data. In this case, the model is effective in explaining car price variations.

2. **Mean Squared Error (MSE):**
    - **Accuracy of Predictions**: The MSE of 5,873,117.46 is a measure of how close the model's predictions are to the actual prices. A lower MSE is desirable, indicating that, on average, your model's predictions are relatively close to the true prices.
    - **Sensitivity to Outliers**: MSE squares the differences between predicted and actual values, making it sensitive to outliers. In the context of car prices, this means the model might be particularly sensitive to extreme price values or errors.
    - **Practical Significance**: When interpreting the MSE, consider the practical significance of the error. For car prices, a MSE of approximately 5.87 million may or may not be acceptable, depending on the specific business or research context. It's crucial to assess whether the level of error is tolerable for this use case.

Overall, the linear regression model demonstrates strong explanatory power (high R-squared) and generally accurate predictions (relatively low MSE). However, when assessing model performance, it's essential to consider domain-specific requirements and business objectives. Additionally, further analysis, such as examining residual plots, can help you gain a more comprehensive understanding of the model's strengths and limitations.