In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
%matplotlib inline
```

In [2]:
```
auto = pd.read_csv('C:/Users/ADMIN/Downloads/Automobile price data _Raw_.csv')
```

In [3]:
```
auto
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compr
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
200	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	8.7
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	173	mpfi	3.58	2.87	8.8
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1	...	145	idi	3.01	3.40	23.0
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5

205 rows × 26 columns

In [4]:
```
new_auto=auto.replace('?' , np.nan)
```

In [5]:
```
new_auto
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compr
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0
1	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0
2	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
200	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	8.7
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	173	mpfi	3.58	2.87	8.8
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1	...	145	idi	3.01	3.40	23.0
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5

205 rows × 26 columns

In [6]:
```
new_auto.describe()
```

	symboling	wheel-base	length	width	height	curb-weight	engine-size	compression-ratio	city-mpg	highway-mpg

	symboling	wheel-base	length	width	height	curb-weight	engine-size	compression-ratio	city-mpg	highway-mpg
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	10.142537	25.219512	30.751220
std	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	3.972040	6.542142	6.886443
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	7.000000	13.000000	16.000000
25%	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	8.600000	19.000000	25.000000
50%	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	9.000000	24.000000	30.000000
75%	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	9.400000	30.000000	34.000000
max	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	23.000000	49.000000	54.000000

In [7]:
```
new_auto.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  164 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       203 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               201 non-null    object 
 19  stroke             201 non-null    object 
 20  compression-ratio  205 non-null    float64
 21  horsepower         203 non-null    object 
 22  peak-rpm           203 non-null    object 
 23  city-mpg           205 non-null    int64  
 24  highway-mpg        205 non-null    int64  
 25  price              201 non-null    object 
dtypes: float64(5), int64(5), object(16)
memory usage: 41.8+ KB
```

In [8]:
```
sns.heatmap(new_auto.isnull(), yticklabels=False,cbar=False,cmap='viridis')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x22222cbfb80>
```

In [9]:
```
new_auto['normalized-losses'] = pd.to_numeric(new_auto['normalized-losses'])

new_auto['normalized-losses'].mean()

new_auto['normalized-losses']=new_auto['normalized-losses'].fillna(value=122)
```

In [10]:
```
new_auto
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compr
0	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0
1	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0
2	1	122.0	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0
3	2	164.0	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0

	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
4	2	164.0	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0
200	-1	95.0	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5
201	-1	95.0	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	8.7
202	-1	95.0	volvo	gas	std	four	sedan	rwd	front	109.1	...	173	mpfi	3.58	2.87	8.8
203	-1	95.0	volvo	diesel	turbo	four	sedan	rwd	front	109.1	...	145	idi	3.01	3.40	23.0
204	-1	95.0	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5

205 rows × 26 columns

In [11]:
```
new_auto['bore'] = pd.to_numeric(new_auto['bore'])

bore=new_auto['bore'].mean()

new_auto['bore']=new_auto['bore'].fillna(value=bore)
```

In [12]:
```
new_auto['bore'].unique()
```

```
array([3.47      , 2.68      , 3.19      , 3.13      , 3.5       ,
       3.31      , 3.62      , 2.91      , 3.03      , 2.97      ,
       3.34      , 3.6       , 2.92      , 3.15      , 3.43      ,
       3.63      , 3.54      , 3.08      , 3.32975124, 3.39      ,
       3.76      , 3.58      , 3.46      , 3.8       , 3.78      ,
       3.17      , 3.35      , 3.59      , 2.99      , 3.33      ,
       3.7       , 3.61      , 3.94      , 3.74      , 2.54      ,
       3.05      , 3.27      , 3.24      , 3.01      ])
```

In [13]:
```
new_auto['stroke'] = pd.to_numeric(new_auto['stroke'])

stroke=new_auto['stroke'].mean()

new_auto['stroke']=new_auto['stroke'].fillna(value=stroke)
```

In [14]:
```
new_auto['horsepower'] = pd.to_numeric(new_auto['horsepower'])

horsepower=new_auto['horsepower'].mean()

new_auto['horsepower']=new_auto['horsepower'].fillna(value=horsepower)
```

In [15]:
```
new_auto['peak-rpm'] = pd.to_numeric(new_auto['peak-rpm'])

peak_rpm=new_auto['peak-rpm'].mean()

new_auto['peak-rpm']=new_auto['peak-rpm'].fillna(value=peak_rpm)
```

In [16]:
```
new_auto['price'] = pd.to_numeric(new_auto['price'])

price=new_auto['price'].mean()

new_auto['price']=new_auto['price'].fillna(value=price)
```

In [17]:
```
sns.heatmap(new_auto.isnull(), yticklabels=False,cbar=False,cmap='viridis')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x22223860340>
```


In [18]:
```
new_auto['num-of-doors'].value_counts()
```

```
four    114
```

```
four    114
two      89
Name: num-of-doors, dtype: int64
```

In [19]:
```
final_data=new_auto.dropna()
```

In [20]:
```
'''
now there is no missing values in data
'''
sns.heatmap(final_data.isnull(), yticklabels=False,cbar=False,cmap='viridis')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x222238ae9d0>
```


In [21]:
```
'''
now we are going to check the pattern and gets some insights
'''
final_data['peak-rpm'].hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x22223977a30>
```


In [22]:
```
from scipy.stats import skew
from scipy.stats import kurtosis
print(skew(final_data['peak-rpm']))
print(kurtosis(final_data['peak-rpm']))
```

```
0.07293872752421213
0.06698801494542161
```

In [23]:
```
sns.barplot(x='normalized-losses',y='make',data=final_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2222369dc10>
```


In [24]:
```
sns.countplot(x='symboling',data= final_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x22223582160>
```


In [25]:
```
sns.countplot(x='fuel-type',data=final_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x222235bdac0>
```


In [26]:
```
sns.countplot(x='aspiration',data=final_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x22223585820>
```

In [27]:
```
sns.countplot(x='num-of-doors',data=final_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2222374daf0>
```

In [28]:
```
sns.countplot(x='drive-wheels',data=final_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x222234e2cd0>
```

In [29]:
```
sns.countplot(x='engine-location',data=final_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2222351b910>
```

In [30]:
```
sns.countplot(x='engine-type',data=final_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2222356f460>
```

In [31]:
```
final_data['num-of-cylinders'].value_counts()
```

```
four      157
six        24
five       11
eight       5
two         4
twelve      1
three       1
Name: num-of-cylinders, dtype: int64
```

In [32]:
```
sns.countplot(x='fuel-system',data=final_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x22223650b80>
```

In [33]:
```
final_data['bore']=pd.to_numeric(final_data['bore'])
final_data['stroke']=pd.to_numeric(final_data['stroke'])
final_data['horsepower']=pd.to_numeric(final_data['horsepower'])
final_data['peak-rpm']=pd.to_numeric(final_data['peak-rpm'])
final_data['price']=pd.to_numeric(final_data['price'])
```

In [34]:

```
final_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 203 entries, 0 to 204
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   symboling          203 non-null    int64  
 1   normalized-losses  203 non-null    float64
 2   make               203 non-null    object 
 3   fuel-type          203 non-null    object 
 4   aspiration         203 non-null    object 
 5   num-of-doors       203 non-null    object 
 6   body-style         203 non-null    object 
 7   drive-wheels       203 non-null    object 
 8   engine-location    203 non-null    object 
 9   wheel-base         203 non-null    float64
 10  length             203 non-null    float64
 11  width              203 non-null    float64
 12  height             203 non-null    float64
 13  curb-weight        203 non-null    int64  
 14  engine-type        203 non-null    object 
 15  num-of-cylinders   203 non-null    object 
 16  engine-size        203 non-null    int64  
 17  fuel-system        203 non-null    object 
 18  bore               203 non-null    float64
 19  stroke             203 non-null    float64
 20  compression-ratio  203 non-null    float64
 21  horsepower         203 non-null    float64
 22  peak-rpm           203 non-null    float64
 23  city-mpg           203 non-null    int64  
 24  highway-mpg        203 non-null    int64  
 25  price              203 non-null    float64
dtypes: float64(11), int64(5), object(10)
memory usage: 52.8+ KB
```

In [35]:
```
new_auto.replace({'num-of-doors': 'four'}, {'num-of-doors': 4}, regex=True)
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compr
0	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0
1	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0
2	1	122.0	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0
3	2	164.0	audi	gas	std	4	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0
4	2	164.0	audi	gas	std	4	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
200	-1	95.0	volvo	gas	std	4	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5
201	-1	95.0	volvo	gas	turbo	4	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	8.7
202	-1	95.0	volvo	gas	std	4	sedan	rwd	front	109.1	...	173	mpfi	3.58	2.87	8.8
203	-1	95.0	volvo	diesel	turbo	4	sedan	rwd	front	109.1	...	145	idi	3.01	3.40	23.0
204	-1	95.0	volvo	gas	turbo	4	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5

205 rows × 26 columns

In [36]:
```
final_data=final_data.replace({'num-of-doors': 'four'}, {'num-of-doors': 4}, regex=True)
```

In [37]:
```
final_data=final_data.replace({'num-of-doors': 'two'}, {'num-of-doors': 2}, regex=True)
```

In [38]:
```
final_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 203 entries, 0 to 204
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   symboling          203 non-null    int64  
 1   normalized-losses  203 non-null    float64
 2   make               203 non-null    object 
 3   fuel-type          203 non-null    object 
 4   aspiration         203 non-null    object 
 5   num-of-doors       203 non-null    int64
```

```
 6   body-style         203 non-null    object
 7   drive-wheels       203 non-null    object
 8   engine-location    203 non-null    object
 9   wheel-base         203 non-null    float64
 10  length             203 non-null    float64
 11  width              203 non-null    float64
 12  height             203 non-null    float64
 13  curb-weight        203 non-null    int64
 14  engine-type        203 non-null    object
 15  num-of-cylinders   203 non-null    object
 16  engine-size        203 non-null    int64
 17  fuel-system        203 non-null    object
 18  bore               203 non-null    float64
 19  stroke             203 non-null    float64
 20  compression-ratio  203 non-null    float64
 21  horsepower         203 non-null    float64
 22  peak-rpm           203 non-null    float64
 23  city-mpg           203 non-null    int64
 24  highway-mpg        203 non-null    int64
 25  price              203 non-null    float64
dtypes: float64(11), int64(6), object(9)
memory usage: 42.8+ KB
```

here we can see that volvo is th safest car for cus and the values which is above 1.5 are not that safe the worst car safety is of porsche

In [39]:
```
a=pd.pivot_table(data=final_data,values=['symboling','price'],index='make') #volvo is the safest car and those are above 1.5 are not that safe
a.sort_values(by='symboling')
```

make	price	symboling
volvo	18063.181818	-1.272727
jaguar	34600.000000	0.000000
mercedes-benz	33647.000000	0.000000
peugot	15489.090909	0.000000
bmw	26118.750000	0.375000
subaru	8541.250000	0.500000
toyota	9885.812500	0.562500
honda	8184.692308	0.615385
isuzu	11061.814677	0.750000
renault	9595.000000	1.000000
plymouth	7963.428571	1.000000
nissan	10415.666667	1.000000
mercury	16503.000000	1.000000
dodge	7790.125000	1.000000
chevrolet	6007.000000	1.000000
mazda	10644.000000	1.187500
audi	17194.589908	1.285714
volkswagen	10077.500000	1.666667
mitsubishi	9239.769231	1.846154
alfa-romero	15498.333333	2.333333
saab	15223.333333	2.500000
porsche	27761.825871	2.600000

In [40]:
```
b=pd.pivot_table(data=final_data,values=['normalized-losses','price'],index='make')
b.sort_values(by='price')
```

make	normalized-losses	price
chevrolet	100.000000	6007.000000
dodge	131.625000	7790.125000
plymouth	128.000000	7963.428571
honda	103.000000	8184.692308
subaru	92.250000	8541.250000

make	normalized-losses	price
mitsubishi	140.615385	9239.769231
renault	122.000000	9595.000000
toyota	110.656250	9885.812500
volkswagen	121.500000	10077.500000
nissan	135.166667	10415.666667
mazda	123.812500	10644.000000
isuzu	122.000000	11061.814677
saab	127.000000	15223.333333
peugot	146.818182	15489.090909
alfa-romero	122.000000	15498.333333
mercury	122.000000	16503.000000
audi	144.285714	17194.589908
volvo	91.454545	18063.181818
bmw	156.000000	26118.750000
porsche	134.800000	27761.825871
mercedes-benz	110.000000	33647.000000
jaguar	129.666667	34600.000000

-the most expensive are jaguar mercedes,porsche,bmw
-also the cars like poorsche which is expensive and provides worst safety
-cars like mistusbhi and peugto have high normalized losses and less safety

In [41]:
```
sns.pairplot(b)
```

```
<seaborn.axisgrid.PairGrid at 0x222239a0940>
```

Questions from data set

1) What kind of safety is provided by which car brand
2)based on price car performance in terms of rpm power
3) which type of car 4wd rwd fwd which type of car is pereferd on the basis of requirements
4)how price is getting affected beacuse of safety and performance
5) on basis of requirements which car will be best suitable

In [42]:
```
'''
here we can say that std engine are less cheaper than turbo
excluding mercedes benz
'''
c=pd.pivot_table(data=final_data,values=['price'],index=['make','aspiration'])
c
```

make	aspiration	price
alfa-romero	std	15498.333333
audi	std	16656.000000
	turbo	18541.064677
bmw	std	26118.750000
chevrolet	std	6007.000000
dodge	std	6900.000000
	turbo	10460.500000
honda	std	8184.692308
isuzu	std	11061.814677
jaguar	std	34600.000000
mazda	std	10644.000000
mercedes-benz	std	38900.000000
	turbo	28394.000000

		price
mercury	turbo	16503.000000
mitsubishi	std	7314.714286
make	aspiration	
	turbo	11485.666667
nissan	std	9869.588235
	turbo	19699.000000
peugot	std	14649.000000
	turbo	16189.166667
plymouth	std	7004.600000
	turbo	10360.500000
porsche	std	27761.825871
renault	std	9595.000000
saab	std	13642.500000
	turbo	18385.000000
subaru	std	7954.200000
	turbo	11476.500000
toyota	std	9859.612903
	turbo	10698.000000
volkswagen	std	9759.000000
	turbo	11670.000000
volvo	std	16197.500000
	turbo	20302.000000

In [135]:
```
pd.pivot_table(data=final_data,values=['price'],index=['make'],columns='fuel-type')
```

	price	
fuel-type	diesel	gas
make		
alfa-romero	NaN	15498.333333
audi	NaN	17194.589908
bmw	NaN	26118.750000
chevrolet	NaN	6007.000000
dodge	NaN	7790.125000
honda	NaN	8184.692308
isuzu	NaN	11061.814677
jaguar	NaN	34600.000000
mazda	18344.000000	10130.666667
mercedes-benz	28394.000000	38900.000000
mercury	NaN	16503.000000
mitsubishi	NaN	9239.769231
nissan	7099.000000	10610.764706
peugot	15797.000000	15232.500000
plymouth	NaN	7963.428571
porsche	NaN	27761.825871
renault	NaN	9595.000000
saab	NaN	15223.333333
subaru	NaN	8541.250000
toyota	8794.666667	9998.689655
volkswagen	9777.500000	10227.500000
volvo	22470.000000	17622.500000

In [43]:
```
#we can say that turbo aspiration cars are mre expensive in most of the case
fig_dims = (8,10)
fig, ax = plt.subplots(figsize=fig_dims)
d=sns.barplot(x='price',y='make',hue='aspiration',ax=ax,data=final_data)
d
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x22224b87b80>
```

In [44]:

```
pd.pivot_table(data=final_data,values=['price'],index=['make','aspiration','num-of-doors'])
```

			price
make	aspiration	num-of-doors	
alfa-romero	std	2	15498.333333
audi	std	2	15250.000000
		4	17007.500000
	turbo	2	13207.129353
		4	23875.000000
bmw	std	2	26238.333333
		4	26047.000000
chevrolet	std	2	5723.000000
		4	6575.000000
dodge	std	2	5974.500000
		4	7362.750000
	turbo	2	10460.500000
honda	std	2	7465.750000
		4	9335.000000
isuzu	std	2	12127.564677
		4	9996.064677
jaguar	std	2	36000.000000
		4	33900.000000
mazda	std	2	9956.111111
		4	11528.428571
mercedes-benz	std	2	40228.000000
		4	37572.000000
	turbo	2	28176.000000
		4	28466.666667
mercury	turbo	2	16503.000000
mitsubishi	std	2	6686.500000
		4	8152.333333
	turbo	2	11927.000000
		4	9279.000000
nissan	std	2	9774.000000
		4	9954.555556
	turbo	2	19699.000000
peugot	std	4	14649.000000
	turbo	4	16189.166667
plymouth	std	2	5572.000000
		4	7362.750000
	turbo	2	10360.500000
porsche	std	2	27761.825871
renault	std	2	9895.000000
		4	9295.000000
saab	std	2	13445.000000
		4	13840.000000
	turbo	2	18150.000000
		4	18620.000000
subaru	std	2	6591.333333
		4	8538.285714
	turbo	4	11476.500000
toyota	std	2	10562.000000
		4	9281.176471
	turbo	4	10698.000000
volkswagen	std	2	9331.250000
		4	10044.166667

		turbo	4	price 000000
volvo	make	aspiration	num-of-doors	16197.500000
		turbo	4	20302.000000

In [45]:
```
fig_dims = (20,7)
fig, ax = plt.subplots(figsize=fig_dims)
sns.lineplot(x='make',y='price',ax=ax,hue='num-of-doors',palette='BrBG',data=final_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x22224c42910>
```

In [46]:
```
pd.pivot_table(data=final_data,values=['price'],index=['body-style'])
```

body-style	price
convertible	21890.500000
hardtop	22208.500000
hatchback	10050.289410
sedan	14534.875093
wagon	12371.960000

In [47]:
```
sns.barplot(x='body-style',y='price',data=final_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x22224a36550>
```

In [48]:
```
sns.barplot(x='drive-wheels',y='price',data=final_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x22224dd3460>
```

In [49]:
```
#we can say that lenght is an important factor as length is increasing the price is also increasi
g
#even wheel base is affecting price
#height is not affecting the price as much
e=pd.pivot_table(data=final_data,values=['price','wheel-base','length','width','height'],index='
make')
e.sort_values(by='length')
```

make	height	length	price	wheel-base	width
chevrolet	52.400000	151.933333	6007.000000	92.466667	62.500000
honda	53.238462	160.769231	8184.692308	94.330769	64.384615
dodge	51.775000	161.450000	7790.125000	95.175000	64.212500
isuzu	52.225000	163.775000	11061.814677	94.825000	63.550000
plymouth	51.971429	164.900000	7963.428571	95.385714	64.271429
mitsubishi	50.692308	168.030769	9239.769231	95.353846	65.253846
subaru	53.750000	168.858333	8541.250000	96.175000	64.950000
alfa-romero	50.000000	169.600000	15498.333333	90.566667	64.566667

make	height	length	price	wheel-base	width
alfa-romero	50.000000	169.600000	15498.333333	90.566667	64.566667
porsche	51.100000	170.260000	27761.825871	92.280000	67.120000
mazda	53.225000	170.368750	10644.000000	96.906250	65.531250
nissan	53.633333	170.988889	10415.666667	95.722222	65.088889
toyota	53.721875	171.934375	9885.812500	98.103125	65.090625
volkswagen	55.183333	172.533333	10077.500000	97.608333	65.616667
mercury	54.800000	178.400000	16503.000000	102.700000	68.000000
renault	52.850000	179.150000	9595.000000	96.100000	66.550000
audi	54.428571	183.828571	17194.589908	102.271429	68.714286
bmw	54.825000	184.500000	26118.750000	103.162500	66.475000
saab	56.100000	186.600000	15223.333333	99.100000	66.500000
volvo	56.236364	188.800000	18063.181818	106.481818	67.963636
peugot	57.181818	191.136364	15489.090909	110.200000	68.390909
mercedes-benz	55.725000	195.262500	33647.000000	110.925000	71.062500
jaguar	51.133333	196.966667	34600.000000	109.333333	69.933333

In [50]:
```
wheel=sns.lineplot(x='wheel-base',y='price',data=final_data)
fig.show(wheel)
```

In [51]:
```
length=sns.lineplot(x='length',y='price',data=final_data)
fig.show(length)
```

In [52]:
```
curb=pd.pivot_table(data=final_data,values=['price','curb-weight'],index='make')
curb.sort_values(by='price')
```

make	curb-weight	price
chevrolet	1757.000000	6007.000000
dodge	2146.375000	7790.125000
plymouth	2220.857143	7963.428571
honda	2096.769231	8184.692308
subaru	2316.250000	8541.250000
mitsubishi	2381.923077	9239.769231
renault	2519.500000	9595.000000
toyota	2441.093750	9885.812500
volkswagen	2343.166667	10077.500000
nissan	2400.388889	10415.666667
mazda	2288.750000	10644.000000
isuzu	2213.500000	11061.814677
saab	2745.500000	15223.333333
peugot	3221.000000	15489.090909
alfa-romero	2639.666667	15498.333333
mercury	2910.000000	16503.000000
audi	2800.714286	17194.589908
volvo	3037.909091	18063.181818
bmw	2929.375000	26118.750000
porsche	2891.200000	27761.825871
mercedes-benz	3696.250000	33647.000000
jaguar	4027.333333	34600.000000

In [53]:
```
final_data['num-of-cylinders']=final_data['num-of-cylinders'].replace(['four','six','eight','two
','three','twelve','five'],[4,6,8,2,3,12,5])
```

In [54]:
```
final_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 203 entries, 0 to 204
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   symboling          203 non-null    int64  
 1   normalized-losses  203 non-null    float64
 2   make               203 non-null    object 
 3   fuel-type          203 non-null    object 
 4   aspiration         203 non-null    object 
 5   num-of-doors       203 non-null    int64  
 6   body-style         203 non-null    object 
 7   drive-wheels       203 non-null    object 
 8   engine-location    203 non-null    object 
 9   wheel-base         203 non-null    float64
 10  length             203 non-null    float64
 11  width              203 non-null    float64
 12  height             203 non-null    float64
 13  curb-weight        203 non-null    int64  
 14  engine-type        203 non-null    object 
 15  num-of-cylinders   203 non-null    int64  
 16  engine-size        203 non-null    int64  
 17  fuel-system        203 non-null    object 
 18  bore               203 non-null    float64
 19  stroke             203 non-null    float64
 20  compression-ratio  203 non-null    float64
 21  horsepower         203 non-null    float64
 22  peak-rpm           203 non-null    float64
 23  city-mpg           203 non-null    int64  
 24  highway-mpg        203 non-null    int64  
 25  price              203 non-null    float64
dtypes: float64(11), int64(7), object(8)
memory usage: 52.8+ KB
```

In [55]:
```
cylinders=pd.pivot_table(data=final_data,values=['price','num-of-cylinders'],index='make')
cylinders.sort_values(by='price')
```

make	num-of-cylinders	price
chevrolet	3.666667	6007.000000
dodge	4.000000	7790.125000
plymouth	4.000000	7963.428571
honda	4.000000	8184.692308
subaru	4.000000	8541.250000
mitsubishi	4.000000	9239.769231
renault	4.000000	9595.000000
toyota	4.250000	9885.812500
volkswagen	4.083333	10077.500000
nissan	4.666667	10415.666667
mazda	3.500000	10644.000000
isuzu	4.000000	11061.814677
saab	4.000000	15223.333333
peugot	4.000000	15489.090909
alfa-romero	4.666667	15498.333333
mercury	4.000000	16503.000000
audi	4.857143	17194.589908
volvo	4.363636	18063.181818
bmw	5.500000	26118.750000
porsche	6.000000	27761.825871
mercedes-benz	6.500000	33647.000000
jaguar	8.000000	34600.000000

In [56]:
```
engine=pd.pivot_table(data=final_data,values=['price','engine-size'],index='make')
engine.sort_values(by='price')
```

	engine-size	price

make	engine-size	price
chevrolet	80.333333	6007.000000
dodge	103.250000	7790.125000
plymouth	106.285714	7963.428571
honda	99.307692	8184.692308
subaru	107.083333	8541.250000
mitsubishi	118.307692	9239.769231
renault	132.000000	9595.000000
toyota	118.812500	9885.812500
volkswagen	107.250000	10077.500000
nissan	127.888889	10415.666667
mazda	101.812500	10644.000000
isuzu	102.500000	11061.814677
saab	121.000000	15223.333333
peugot	135.818182	15489.090909
alfa-romero	137.333333	15498.333333
mercury	140.000000	16503.000000
audi	130.714286	17194.589908
volvo	142.272727	18063.181818
bmw	166.875000	26118.750000
porsche	187.200000	27761.825871
mercedes-benz	226.500000	33647.000000
jaguar	280.666667	34600.000000

In [57]:
```
fuel=pd.pivot_table(data=final_data,values=['price'],index='fuel-system')
fuel.sort_values(by='price')
```

fuel-system	price
1bbl	7555.545455
2bbl	7608.170586
spdi	10990.444444
spfi	11048.000000
4bbl	12145.000000
mfi	12964.000000
idi	16103.578947
mpfi	17607.841491

In [58]:
```
performance=pd.pivot_table(data=final_data,values=['price','bore','stroke','compression-ratio'],
index=['make'])
performance.sort_values(by='price')
```

make	bore	compression-ratio	price	stroke
chevrolet	2.990000	9.566667	6007.000000	3.083333
dodge	3.102500	8.763750	7790.125000	3.362500
plymouth	3.121429	8.671429	7963.428571	3.375714
honda	3.021538	9.215385	8184.692308	3.462308
subaru	3.620000	8.816667	8541.250000	2.616667
mitsubishi	3.250769	8.061538	9239.769231	3.493846
renault	3.460000	8.700000	9595.000000	3.900000
toyota	3.280000	10.340625	9885.812500	3.255000
volkswagen	3.130000	13.625000	10077.500000	3.400000
nissan	3.254444	9.794444	10415.666667	3.313333
mazda	3.291188	9.725000	10644.000000	3.282606
isuzu	3.200000	9.225000	11061.814677	3.170000
saab	3.373333	9.201667	15223.333333	2.903333
peugot	3.582727	14.000000	15489.090909	3.160000

make	bore	compression-ratio	price	stroke
alfa-romero	3.205667	9.000000	15498.333333	2.943333
mercury	3.780000	8.000000	16503.000000	3.120000
audi	3.172857	8.400000	17194.589908	3.400000
volvo	3.662727	10.227273	18063.181818	3.147273
bmw	3.473750	8.575000	26118.750000	3.167500
porsche	3.820000	9.600000	27761.825871	2.984000
mercedes-benz	3.605000	14.825000	33647.000000	3.432500
jaguar	3.600000	9.233333	34600.000000	3.700000

In [59]:

```
#horsepower is much more for premium cars
power =pd.pivot_table(data=final_data,values=['price','horsepower','peak-rpm'],index=['make'])
power.sort_values(by='price')
```

make	horsepower	peak-rpm	price
chevrolet	62.666667	5300.000000	6007.000000
dodge	84.375000	5375.000000	7790.125000
plymouth	86.714286	5357.142857	7963.428571
honda	80.230769	5753.846154	8184.692308
subaru	86.250000	4775.000000	8541.250000
mitsubishi	104.076923	5269.230769	9239.769231
renault	104.256158	5125.369458	9595.000000
toyota	92.781250	4859.375000	9885.812500
volkswagen	81.083333	5154.166667	10077.500000
nissan	102.555556	5177.777778	10415.666667
mazda	86.875000	5137.500000	10644.000000
isuzu	77.000000	5150.000000	11061.814677
saab	126.666667	5333.333333	15223.333333
peugot	99.818182	4668.181818	15489.090909
alfa-romero	125.333333	5000.000000	15498.333333
mercury	175.000000	5000.000000	16503.000000
audi	121.000000	5500.000000	17194.589908
volvo	128.000000	5290.909091	18063.181818
bmw	138.875000	5068.750000	26118.750000
porsche	210.400000	5790.000000	27761.825871
mercedes-benz	146.250000	4487.500000	33647.000000
jaguar	204.666667	4833.333333	34600.000000

In [62]:

```
'''
premium cars are having lesser mileage in both city and highway
'''

mileage=pd.pivot_table(data=final_data,values=['price','city-mpg','highway-mpg'],index=['make'])
mileage.sort_values(by='price')
```

make	city-mpg	highway-mpg	price
chevrolet	41.000000	46.333333	6007.000000
dodge	28.500000	34.625000	7790.125000
plymouth	28.142857	34.142857	7963.428571
honda	30.384615	35.461538	8184.692308
subaru	26.333333	30.750000	8541.250000
mitsubishi	24.923077	31.153846	9239.769231
renault	23.000000	31.000000	9595.000000
toyota	27.500000	32.906250	9885.812500
volkswagen	28.583333	34.916667	10077.500000
nissan	27.000000	32.944444	10415.666667
mazda	25.062500	31.312500	10644.000000

make	city-mpg	highway-mpg	price
mazda	25.062500	31.312500	10644.000000
isuzu	31.000000	36.000000	11061.814677
saab	20.333333	27.333333	15223.333333
peugot	22.454545	26.636364	15489.090909
alfa-romero	20.333333	26.666667	15498.333333
mercury	19.000000	24.000000	16503.000000
audi	18.857143	24.142857	17194.589908
volvo	21.181818	25.818182	18063.181818
bmw	19.375000	25.375000	26118.750000
porsche	17.400000	26.000000	27761.825871
mercedes-benz	18.500000	21.000000	33647.000000
jaguar	14.333333	18.333333	34600.000000

now will do some multivariant analysis on data for better understanding

In [97]:
```
#from this tbale we can say that gas cars are much more stable
#and also gas are much power in terms of horsepower
#so one who is looking for a horsepower based he can choose fuel type as gas

f=pd.pivot_table(data=final_data,columns=['fuel-type'],values=['price','symboling','length','horsepower','num-of-cylinders','peak-rpm'],index=['make'])
f
```

	horsepower		length		num-of-cylinders		peak-rpm		price		symboling	
fuel-type	diesel	gas	diesel	gas	diesel	gas	diesel	gas	diesel	gas	diesel	gas
make												
alfa-romero	NaN	125.333333	NaN	169.600000	NaN	4.666667	NaN	5000.000000	NaN	15498.333333	NaN	2.333333
audi	NaN	121.000000	NaN	183.828571	NaN	4.857143	NaN	5500.000000	NaN	17194.589908	NaN	1.285714
bmw	NaN	138.875000	NaN	184.500000	NaN	5.500000	NaN	5068.750000	NaN	26118.750000	NaN	0.375000
chevrolet	NaN	62.666667	NaN	151.933333	NaN	3.666667	NaN	5300.000000	NaN	6007.000000	NaN	1.000000
dodge	NaN	84.375000	NaN	161.450000	NaN	4.000000	NaN	5375.000000	NaN	7790.125000	NaN	1.000000
honda	NaN	80.230769	NaN	160.769231	NaN	4.000000	NaN	5753.846154	NaN	8184.692308	NaN	0.615385
isuzu	NaN	77.000000	NaN	163.775000	NaN	4.000000	NaN	5150.000000	NaN	11061.814677	NaN	0.750000
jaguar	NaN	204.666667	NaN	196.966667	NaN	8.000000	NaN	4833.333333	NaN	34600.000000	NaN	0.000000
mazda	72.000000	87.866667	175.000	170.060000	4.0	3.466667	4200.0	5200.000000	18344.000000	10130.666667	0.000000	1.266667
mercedes-benz	123.000000	169.500000	192.975	197.550000	5.0	8.000000	4350.0	4625.000000	28394.000000	38900.000000	-0.750000	0.750000
mercury	NaN	175.000000	NaN	178.400000	NaN	4.000000	NaN	5000.000000	NaN	16503.000000	NaN	1.000000
mitsubishi	NaN	104.076923	NaN	168.030769	NaN	4.000000	NaN	5269.230769	NaN	9239.769231	NaN	1.846154
nissan	55.000000	105.352941	165.300	171.323529	4.0	4.705882	4800.0	5200.000000	7099.000000	10610.764706	1.000000	1.000000
peugot	95.000000	103.833333	191.580	190.766667	4.0	4.000000	4150.0	5100.000000	15797.000000	15232.500000	0.000000	0.000000
plymouth	NaN	86.714286	NaN	164.900000	NaN	4.000000	NaN	5357.142857	NaN	7963.428571	NaN	1.000000
porsche	NaN	210.400000	NaN	170.260000	NaN	6.000000	NaN	5790.000000	NaN	27761.825871	NaN	2.600000
renault	NaN	104.256158	NaN	179.150000	NaN	4.000000	NaN	5125.369458	NaN	9595.000000	NaN	1.000000
saab	NaN	126.666667	NaN	186.600000	NaN	4.000000	NaN	5333.333333	NaN	15223.333333	NaN	2.500000
subaru	NaN	86.250000	NaN	168.858333	NaN	4.000000	NaN	4775.000000	NaN	8541.250000	NaN	0.500000
toyota	61.666667	96.000000	169.400	172.196552	4.0	4.275862	4500.0	4896.551724	8794.666667	9998.689655	-0.333333	0.655172
volkswagen	60.000000	91.625000	173.825	171.887500	4.0	4.125000	4650.0	5406.250000	9777.500000	10227.500000	1.500000	1.750000
volvo	106.000000	130.200000	188.800	188.800000	6.0	4.200000	4800.0	5340.000000	22470.000000	17622.500000	-1.000000	-1.300000

In [96]:
```
plt.figure(figsize = (14,14))
plt.subplot(221)
plt.title('correlation')
sns.scatterplot(final_data['horsepower'],final_data['peak-rpm'])

plt.subplot(222)
plt.title('maker vs no of cylinders')
sns.barplot(x='num-of-cylinders',y='make',data=final_data)

plt.subplot(223)
plt.title('horsepower vs fuel type')
```

```
sns.barplot(x='fuel-type',y='horsepower',data=final_data)

plt.subplot(224)
plt.title('distribution')
sns.distplot(final_data['length'])
plt.savefig('Dashboard.png')
```

In [104]:

```
#here we can engine size i expontentially affecting the price of the car as thee engine size is i
ncreasing the price of car is also increasing drastically
#even horsepower is dependent on engine size more the size more the horsepower is most of the cas
e
#wheel base is not impacting much price so there is no as such a proper relation

g= pd.pivot_table(data=final_data,values=['wheel-base','length','engine-size','horsepower','pric
e','symboling'],index=['make'])
g.sort_values(by='wheel-base')
```

make	engine-size	horsepower	length	price	symboling	wheel-base
alfa-romero	137.333333	125.333333	169.600000	15498.333333	2.333333	90.566667
porsche	187.200000	210.400000	170.260000	27761.825871	2.600000	92.280000
chevrolet	80.333333	62.666667	151.933333	6007.000000	1.000000	92.466667
honda	99.307692	80.230769	160.769231	8184.692308	0.615385	94.330769
isuzu	102.500000	77.000000	163.775000	11061.814677	0.750000	94.825000
dodge	103.250000	84.375000	161.450000	7790.125000	1.000000	95.175000
mitsubishi	118.307692	104.076923	168.030769	9239.769231	1.846154	95.353846
plymouth	106.285714	86.714286	164.900000	7963.428571	1.000000	95.385714
nissan	127.888889	102.555556	170.988889	10415.666667	1.000000	95.722222
renault	132.000000	104.256158	179.150000	9595.000000	1.000000	96.100000
subaru	107.083333	86.250000	168.858333	8541.250000	0.500000	96.175000
mazda	101.812500	86.875000	170.368750	10644.000000	1.187500	96.906250
volkswagen	107.250000	81.083333	172.533333	10077.500000	1.666667	97.608333
toyota	118.812500	92.781250	171.934375	9885.812500	0.562500	98.103125
saab	121.000000	126.666667	186.600000	15223.333333	2.500000	99.100000
audi	130.714286	121.000000	183.828571	17194.589908	1.285714	102.271429
mercury	140.000000	175.000000	178.400000	16503.000000	1.000000	102.700000
bmw	166.875000	138.875000	184.500000	26118.750000	0.375000	103.162500
volvo	142.272727	128.000000	188.800000	18063.181818	-1.272727	106.481818
jaguar	280.666667	204.666667	196.966667	34600.000000	0.000000	109.333333
peugot	135.818182	99.818182	191.136364	15489.090909	0.000000	110.200000
mercedes-benz	226.500000	146.250000	195.262500	33647.000000	0.000000	110.925000

In [107]:

```
h=pd.pivot_table(data=final_data,values=['price','wheel-base','num-of-doors'],index='make')
h.sort_values(by='price')
```

make	num-of-doors	price	wheel-base
chevrolet	2.666667	6007.000000	92.466667
dodge	3.000000	7790.125000	95.175000
plymouth	3.142857	7963.428571	95.385714
honda	2.769231	8184.692308	94.330769
subaru	3.500000	8541.250000	96.175000
mitsubishi	2.615385	9239.769231	95.353846
renault	3.000000	9595.000000	96.100000
toyota	3.125000	9885.812500	98.103125
volkswagen	3.333333	10077.500000	97.608333

make	num-of-doors	price	wheel-base
nissan	3.000000	10415.666667	95.722222
mazda	2.875000	10644.000000	96.906250
isuzu	3.000000	11061.814677	94.825000
saab	3.000000	15223.333333	99.100000
peugot	4.000000	15489.090909	110.200000
alfa-romero	2.000000	15498.333333	90.566667
mercury	2.000000	16503.000000	102.700000
audi	3.428571	17194.589908	102.271429
volvo	4.000000	18063.181818	106.481818
bmw	3.250000	26118.750000	103.162500
porsche	2.000000	27761.825871	92.280000
mercedes-benz	3.250000	33647.000000	110.925000
jaguar	3.333333	34600.000000	109.333333

In [113]:

```
#std engine gives better mileage and reduce car cost
# horsepower of turbo engine is more
# only mercedes benz is having expensive std based beacuse of the horsepower is offere more than
turbo car
# here we can say l is the cheapest engine type and ohcv and dohcv are expensive fuel type and be
acuse of this two engine type the price goes up drastically
i = pd.pivot_table(data=final_data,values=['price','city-mpg','highway-mpg','horsepower'],index=
['make','engine-type'])
i
```

make	engine-type	city-mpg	highway-mpg	horsepower	price
alfa-romero	dohc	21.000000	27.000000	111.000000	14997.500000
	ohcv	19.000000	26.000000	154.000000	16500.000000
audi	ohc	18.857143	24.142857	121.000000	17194.589908
bmw	ohc	19.375000	25.375000	138.875000	26118.750000
chevrolet	l	47.000000	53.000000	48.000000	5151.000000
	ohc	38.000000	43.000000	70.000000	6435.000000
dodge	ohc	28.500000	34.625000	84.375000	7790.125000
honda	ohc	30.384615	35.461538	80.230769	8184.692308
isuzu	ohc	31.000000	36.000000	77.000000	11061.814677
jaguar	dohc	15.000000	19.000000	176.000000	33900.000000
	ohcv	13.000000	17.000000	262.000000	36000.000000
mazda	ohc	27.833333	34.083333	79.333333	9852.000000
	rotor	16.750000	23.000000	109.500000	13020.000000
mercedes-benz	ohc	22.000000	25.000000	123.000000	28394.000000
	ohcv	15.000000	17.000000	169.500000	38900.000000
mercury	ohc	19.000000	24.000000	175.000000	16503.000000
mitsubishi	ohc	24.923077	31.153846	104.076923	9239.769231
nissan	ohc	31.500000	37.583333	72.500000	7565.666667
	ohcv	18.000000	23.666667	162.666667	16115.666667
peugot	l	22.454545	26.636364	99.818182	15489.090909
plymouth	ohc	28.142857	34.142857	86.714286	7963.428571
porsche	dohcv	17.000000	28.000000	288.000000	13207.129353
	ohc	19.000000	27.000000	143.000000	22018.000000
	ohcf	17.000000	25.000000	207.000000	34528.000000
renault	ohc	23.000000	31.000000	104.256158	9595.000000
saab	dohc	19.000000	26.000000	160.000000	18385.000000
	ohc	21.000000	28.000000	110.000000	13642.500000
subaru	ohcf	26.333333	30.750000	86.250000	8541.250000
toyota	dohc	21.666667	25.666667	143.000000	13805.333333
	ohc	28.846154	34.576923	81.192308	8981.307692
volkswagen	ohc	28.583333	34.916667	81.083333	10077.500000
volvo	ohc	21.500000	26.100000	127.400000	17721.000000
	ohcv	18.000000	23.000000	134.000000	21485.000000

city-mpg highway-mpg horsepower price

In [118]:

```
# we can say that length and width are most important in price prediction as the length na dwidth
both increases simulataneiously the price also increase and convertibel and hardtop are expensive
s body style
pivot_table(data=final_data,values=['height','length','width','price'],index=['make','body-style
'])
```

make	body-style	height	length	price	width
alfa-romero	convertible	48.800000	168.800000	14997.500000	64.100000
	hatchback	52.400000	171.200000	16500.000000	65.500000
audi	hatchback	52.000000	178.200000	13207.129353	67.900000
	sedan	54.660000	183.180000	17647.000000	68.340000
	wagon	55.700000	192.700000	18920.000000	71.400000
bmw	sedan	54.825000	184.500000	26118.750000	66.475000
chevrolet	hatchback	52.600000	148.500000	5723.000000	61.950000
	sedan	52.000000	158.800000	6575.000000	63.600000
dodge	hatchback	50.640000	160.480000	7819.800000	64.300000
	sedan	50.600000	157.300000	7150.500000	63.800000
	wagon	59.800000	174.600000	8921.000000	64.600000
honda	hatchback	52.285714	153.457143	7054.428571	64.314286
	sedan	53.560000	171.740000	9945.000000	64.580000
	wagon	58.300000	157.100000	7295.000000	63.900000
isuzu	hatchback	51.400000	172.600000	11048.000000	65.200000
	sedan	52.500000	160.833333	11066.419569	63.000000
jaguar	sedan	51.133333	196.966667	34600.000000	69.933333
mazda	hatchback	52.360000	168.670000	10085.000000	65.490000
	sedan	54.666667	173.200000	11575.666667	65.600000
mercedes-benz	convertible	50.800000	180.300000	35056.000000	70.500000
	hardtop	55.150000	193.350000	36788.000000	71.150000
	sedan	56.500000	201.050000	33074.000000	71.350000
	wagon	58.700000	190.900000	28248.000000	70.300000
mercury	hatchback	54.800000	178.400000	16503.000000	68.000000
mitsubishi	hatchback	50.288889	166.088889	9597.888889	65.188889
	sedan	51.600000	172.400000	8434.000000	65.400000
nissan	hardtop	53.300000	162.400000	8249.000000	63.800000
	hatchback	51.420000	171.780000	14409.000000	66.540000
	sedan	54.655556	170.166667	8604.555556	64.555556
	wagon	54.366667	175.000000	9915.666667	64.700000
peugot	sedan	56.600000	186.700000	15758.571429	68.385714
	wagon	58.200000	198.900000	15017.500000	68.400000
plymouth	hatchback	50.600000	161.275000	8130.500000	64.425000
	sedan	50.800000	167.300000	7150.500000	63.800000
	wagon	59.800000	174.600000	8921.000000	64.600000
porsche	convertible	51.600000	168.900000	37028.000000	65.000000
	hardtop	51.600000	168.900000	33278.000000	65.000000
	hatchback	50.350000	172.300000	17612.564677	70.300000
renault	hatchback	50.500000	176.800000	9895.000000	66.600000
	wagon	55.200000	181.500000	9295.000000	66.500000
saab	hatchback	56.100000	186.600000	15013.333333	66.500000
	sedan	56.100000	186.600000	15433.333333	66.500000
subaru	hatchback	54.366667	157.366667	6591.333333	63.600000
	sedan	53.220000	172.000000	9070.600000	65.400000
	wagon	53.950000	173.550000	9342.000000	65.400000
toyota	convertible	53.000000	176.200000	17669.000000	65.600000
	hardtop	52.000000	176.200000	9762.333333	65.600000
	hatchback	53.064286	170.214286	9616.000000	65.114286
	sedan	53.580000	171.720000	9542.200000	65.160000
	wagon	57.850000	174.225000	9836.000000	64.325000

make	body-style	height	length	price	width
volkswagen	convertible	[illegible]	[illegible]	[illegible]	[illegible]
	hatchback	51.400000	165.700000	9980.000000	64.000000
	sedan	55.566667	173.588889	9673.888889	65.811111
	wagon	55.100000	183.100000	12290.000000	66.900000
volvo	sedan	55.762500	188.800000	18726.875000	68.250000
	wagon	57.500000	188.800000	16293.333333	67.200000

In [138]:

```
#mpfi based fuel system are expensive and 2bbl are cheapest we can get on low price
pd.pivot_table(data=final_data,values=['price'],index=['make','drive-wheels','fuel-system'])
```

make	drive-wheels	fuel-system	price
alfa-romero	rwd	mpfi	15498.333333
audi	4wd	mpfi	15328.564677
	fwd	mpfi	17941.000000
bmw	rwd	mpfi	26118.750000
chevrolet	fwd	2bbl	6007.000000
dodge	fwd	2bbl	6900.000000
		mfi	12964.000000
		mpfi	7957.000000
honda	fwd	1bbl	7555.545455
		2bbl	10345.000000
		mpfi	12945.000000
isuzu	fwd	2bbl	13207.129353
	rwd	2bbl	6785.000000
		spfi	11048.000000
jaguar	rwd	mpfi	34600.000000
mazda	fwd	2bbl	8160.000000
	rwd	4bbl	12145.000000
		idi	18344.000000
		mpfi	16962.500000
mercedes-benz	rwd	idi	28394.000000
		mpfi	38900.000000
mercury	rwd	mpfi	16503.000000
mitsubishi	fwd	2bbl	6987.333333
		spdi	11170.428571
nissan	fwd	2bbl	7608.090909
		idi	7099.000000
		mpfi	13799.000000
	rwd	mpfi	18432.333333
peugot	rwd	idi	15797.000000
		mpfi	15232.500000
plymouth	fwd	2bbl	7004.600000
		spdi	7957.000000
	rwd	spdi	12764.000000
porsche	rwd	mpfi	27761.825871
renault	fwd	mpfi	9595.000000
saab	fwd	mpfi	15223.333333
subaru	4wd	2bbl	8283.000000
		mpfi	11476.500000
	fwd	2bbl	6907.000000
		mpfi	10079.000000
toyota	4wd	2bbl	8338.000000
	fwd	2bbl	7175.777778
		idi	8794.666667
		mpfi	10270.500000
	rwd	2bbl	8148.000000
		mpfi	12610.500000
volkswagen	fwd	idi	9777.500000

				price
			mpfi	10227.500000
volvo	make	drive-wheels	fuel-system	22470.000000
		rwd	idi	
			mpfi	17622.500000

In [128]:
```
final_data_copy=final_data.copy()
final_data_copy
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compr
0	3	122.0	alfa-romero	gas	std	2	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0
1	3	122.0	alfa-romero	gas	std	2	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0
2	1	122.0	alfa-romero	gas	std	2	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0
3	2	164.0	audi	gas	std	4	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0
4	2	164.0	audi	gas	std	4	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
200	-1	95.0	volvo	gas	std	4	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5
201	-1	95.0	volvo	gas	turbo	4	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	8.7
202	-1	95.0	volvo	gas	std	4	sedan	rwd	front	109.1	...	173	mpfi	3.58	2.87	8.8
203	-1	95.0	volvo	diesel	turbo	4	sedan	rwd	front	109.1	...	145	idi	3.01	3.40	23.0
204	-1	95.0	volvo	gas	turbo	4	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5

203 rows × 26 columns

In [131]:
```
fianl_data_copy1=final_data_copy.drop(['city-mpg','highway-mpg'],axis=1)
```

In [132]:
```
fianl_data_copy1
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-type	num-of-cylinders	engine-size	fuel-system
0	3	122.0	alfa-romero	gas	std	2	convertible	rwd	front	88.6	...	dohc	4	130	mpfi
1	3	122.0	alfa-romero	gas	std	2	convertible	rwd	front	88.6	...	dohc	4	130	mpfi
2	1	122.0	alfa-romero	gas	std	2	hatchback	rwd	front	94.5	...	ohcv	6	152	mpfi
3	2	164.0	audi	gas	std	4	sedan	fwd	front	99.8	...	ohc	4	109	mpfi
4	2	164.0	audi	gas	std	4	sedan	4wd	front	99.4	...	ohc	5	136	mpfi
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
200	-1	95.0	volvo	gas	std	4	sedan	rwd	front	109.1	...	ohc	4	141	mpfi
201	-1	95.0	volvo	gas	turbo	4	sedan	rwd	front	109.1	...	ohc	4	141	mpfi
202	-1	95.0	volvo	gas	std	4	sedan	rwd	front	109.1	...	ohcv	6	173	mpfi
203	-1	95.0	volvo	diesel	turbo	4	sedan	rwd	front	109.1	...	ohc	6	145	idi
204	-1	95.0	volvo	gas	turbo	4	sedan	rwd	front	109.1	...	ohc	4	141	mpfi

203 rows × 24 columns

In [133]:
```
corr=fianl_data_copy1.corr()
corr
```

	symboling	normalized-losses	num-of-doors	wheel-base	length	width	height	curb-weight	num-of-cylinders	engine-size	bore	stroke	compressio ra
symboling	1.000000	0.465972	-0.664550	-0.532929	-0.357856	-0.232241	-0.541323	-0.228520	-0.114266	-0.105985	-0.129194	-0.007565	-0.172343
normalized-losses	0.465972	1.000000	-0.362716	-0.053305	0.024851	0.088535	-0.368117	0.100921	0.109761	0.114135	-0.024915	0.053323	-0.115170

	symboling	normalized-losses	num-of-doors	wheel-base	length	width	height	curb-weight	num-of-cylinders	engine-size	bore	stroke	compressio ra
num-of-doors	-0.664550	0.362716	1.000000	0.445696	0.393593	0.202072	0.547651	0.195683	-0.016530	0.017519	0.113117	-0.010954	0.165799
wheel-base	-0.532929	-0.053305	0.445696	1.000000	0.874651	0.794488	0.588281	0.775870	0.338853	0.568141	0.486549	0.163145	0.254105
length	-0.357856	0.024851	0.393593	0.874651	1.000000	0.840300	0.486316	0.878719	0.431155	0.682984	0.603505	0.132512	0.154219
width	-0.232241	0.088535	0.202072	0.794488	0.840300	1.000000	0.274216	0.867307	0.545508	0.735016	0.556674	0.185083	0.178893
height	-0.541323	-0.368117	0.547651	0.588281	0.486316	0.274216	1.000000	0.293892	-0.015047	0.063622	0.164750	-0.054517	0.253871
curb-weight	-0.228520	0.100921	0.195683	0.775870	0.878719	0.867307	0.293892	1.000000	0.609312	0.850236	0.647792	0.171112	0.156838
num-of-cylinders	-0.114266	0.109761	-0.016530	0.338853	0.431155	0.545508	-0.015047	0.609312	1.000000	0.846170	0.230763	0.009662	-0.016033
engine-size	-0.105985	0.114135	0.017519	0.568141	0.682984	0.735016	0.063622	0.850236	0.846170	1.000000	0.582682	0.205257	0.029468
bore	-0.129194	-0.024915	0.113117	0.486549	0.603505	0.556674	0.164750	0.647792	0.230763	0.582682	1.000000	-0.054250	-0.001715
stroke	-0.007565	0.053323	-0.010654	0.163145	0.132512	0.185083	-0.054517	0.171112	0.009662	0.205257	-0.054250	1.000000	0.185727
compression-ratio	-0.172343	-0.115170	0.165799	0.254105	0.154219	0.178893	0.253871	0.156838	-0.016033	0.029468	-0.001715	0.185727	1.000000
horsepower	0.068321	0.204549	-0.124320	0.353257	0.559644	0.646594	-0.107730	0.752598	0.691533	0.812989	0.579819	0.090865	-0.195626
peak-rpm	0.271274	0.235915	-0.241249	-0.359484	-0.282785	-0.216138	-0.315047	-0.266052	-0.125298	-0.243768	-0.251197	-0.066720	-0.430838
price	-0.083020	0.136770	0.046051	0.582379	0.683493	0.728917	0.132613	0.820439	0.687381	0.861545	0.531659	0.084171	0.075992

In [134]:

```
# now at last getting the correlation of the data
# so now will plot a heatmap for getting the correlation

plt.figure(figsize=(20,15))
sns.heatmap(corr, annot = True)
plt.show()
```

final report

1) symboling affects the price as the safety increases the price of car is also increasing
2) gas type of cars are more powerfull and stable
3) std based engine are less cheaper and turbo are more expensive even std cars give a better mileage and turbo does not
4) engine size is expontentially affecting the price of the car as the engine size is increasing the price of car is also increasing drastically even horsepower is dependent on engine size more the size more the horsepower is most of the case
5)wheel base is not impacting much price so there is no as such a proper relation
6) in terms of body style most expensive styles are convertible and hardtop style and also some premium brand sedans style is also affecting the price a lot
7)length and width shows a strong correlation beacuse when length and width of a car increases simultaneously the price also increases
8)horsepower of turbo engine is more .Only mercedes benz is having expensive std based beacuse of the horsepower is offere more than turbo car here we can say l is the cheapest engine type and ohcv and dohcv are expensive fuel type and beacuse of this two engine type the price goes up drastically
9)mpfi based fuel system are expensive and 2bbl are cheapest we can get on low price

END