Domain: Automobile
Project 04: Automobile data
This dataset contains information about cars
Attribute Information:

Attribute	**Attribute Range**
----------------------	-------------------------
1. symboling	-3, -2, -1, 0, 1, 2, 3.
2. normalized-losses	continuous from 65 to 256.
3. make	alfa-romero, audi, bmw, chevrolet,dodge,honda, isuzu, jaguar, mazda, mercedes-benz, mercury, mitsubishi,nissan,peugot,plymouth,porsche, renault, saab, subaru, toyota, volkswagen volvo
4. fuel-type	diesel, gas.
5. aspiration	std, turbo.
6. num-of-doors	four, two.
7. body-style	hardtop, wagon, sedan, hatchback, convertible.
8. drive-wheels	4wd, fwd, rwd.
9. engine-location	front, rear.
10. wheel-base	continuous from 86.6120.9.
11. length	continuous from 141.1 to 208.1.
12. width	continuous from 60.3 to 72.3.
13. height	continuous from 47.8 to 59.8.
14. curb-weight	continuous from 1488 to 4066.
15. engine-type	dohc,dohcv,l,ohc,ohcf,ohcv,rotor.
16. number-of-cylinders	eight,five,four,six,three,twelve,two.
17. engine-size	continuous from 61 to 326.
18. fuel-system	1bbl, 2bbl, 4bbl, idi, mfi, mpfi, spdi, spfi.
19. bore	continuous from 2.54 to 3.94.
20. stroke	continuous from 2.07 to 4.17.
21. compression-ratio	continuous from 7 to 23.
22. horsepower	continuous from 48 to 288.
23. peak-rpm	continuous from 4150 to 6600
24. city-mpg	continuous from 13 to 49.
25. highway-mpg	continuous from 16 to 54.
26. price	continuous from 5118 to 45400.

Exploration ideas
Loading and cleaning data.
Variable analysis to see its impact on automobile pricing.
Summary Statistics of different variables.
Univariate and bivariate analysis
Make, Curb-weight, Drive wheels analysis.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import pandas_profiling
import collections
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [64]:

```
# Print multiple statements in same line
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

Loading and Cleaning Data

In [65]:

```
automobile=pd.read_csv("Automobile_data.txt", sep = ',')
automobile.head()
```

Out[65]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0

5 rows × 26 columns

In [66]:

```
report = pandas_profiling.ProfileReport(automobile)
# convert profile report as html file
#report.to_file("automobile.html")
```

Summary Statistics of different variables

In [67]:

```
automobile.dtypes
```

Out[67]:

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

In [68]:

```
#automobile.info()
automobile.describe(include='all')
```

Out[68]:

	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 ratio
count	205.000000	205	205	205	205	205	205	205	205	205.000000	...	205.000000	205	205	205	205.00
unique	NaN	52	22	2	2	3	5	3	2	NaN	...	NaN	8	39	37	NaN
top	NaN	?	toyota	gas	std	four	sedan	fwd	front	NaN	...	NaN	mpfi	3.62	3.40	NaN
freq	NaN	41	32	185	168	114	96	120	202	NaN	...	NaN	94	23	20	NaN
mean	0.834146	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	98.756585	...	126.907317	NaN	NaN	NaN	10.142
std	1.245307	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	6.021776	...	41.642693	NaN	NaN	NaN	3.9720
min	-2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	86.600000	...	61.000000	NaN	NaN	NaN	7.0000
25%	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	94.500000	...	97.000000	NaN	NaN	NaN	8.6000
50%	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	97.000000	...	120.000000	NaN	NaN	NaN	9.0000
75%	2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	102.400000	...	141.000000	NaN	NaN	NaN	9.4000
max	3.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	120.900000	...	326.000000	NaN	NaN	NaN	23.000

11 rows × 26 columns

In [69]:

```
automobile.isnull().sum()
```

Out[69]:

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

dtype: int64

In [70]:
```
# Find out number of records having '?' value for normalized losses
automobile['normalized-losses'].loc[automobile['normalized-losses'] == '?'].count()
```

Out[70]:
41

In [71]:
```
# Setting the missing value to mean of normalized losses and conver the datatype to integer
nl = automobile['normalized-losses'].loc[automobile['normalized-losses'] != '?']
nlmean = nl.astype(str).astype(int).mean()
automobile['normalized-losses'] = automobile['normalized-losses'].replace('?',nlmean).astype(int)
automobile['normalized-losses'].head()
```

Out[71]:
```
0    122
1    122
2    122
3    164
4    164
Name: normalized-losses, dtype: int32
```

In [72]:
```
# Find out the number of values which are not numeric
automobile['price'].str.isnumeric().value_counts()
```

Out[72]:
```
True     201
False      4
Name: price, dtype: int64
```

In [73]:
```
# List out the values which are not numeric
automobile['price'].loc[automobile['price'].str.isnumeric() == False]
```

Out[73]:
```
9      ?
44     ?
45     ?
129    ?
Name: price, dtype: object
```

In [74]:
```
#Setting the missing value to mean of price and convert the datatype to integer
price = automobile['price'].loc[automobile['price'] != '?']
pmean = price.astype(str).astype(int).mean()
automobile['price'] = automobile['price'].replace('?',pmean).astype(int)
automobile['price'].head()
```

Out[74]:
```
0    13495
1    16500
2    16500
3    13950
4    17450
Name: price, dtype: int32
```

In [75]:
```
# Checking the numberic and replacing with mean value and conver the datatype to integer
automobile['horsepower'].str.isnumeric().value_counts()
horsepower = automobile['horsepower'].loc[automobile['horsepower'] != '?']
hpmean = horsepower.astype(str).astype(int).mean()
automobile['horsepower'] = automobile['horsepower'].replace('?',pmean).astype(int)
```

Out[75]:
```
True     203
False      2
Name: horsepower, dtype: int64
```

In [76]:
```
automobile.loc[automobile['horsepower']].quantile(1.00)
```

Out[76]:
```
symboling              3.0
normalized-losses    231.0
wheel-base           120.9
length               208.1
width                 71.7
height                59.8
curb-weight         4066.0
engine-size          308.0
compression-ratio     23.0
horsepower           184.0
city-mpg              45.0
highway-mpg           50.0
```

```
price            40960.0
Name: 1.0, dtype: float64
```

In [77]:

```
automobile.describe(include='all')
```

Out[77]:

	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 ratio
count	205.000000	205.000000	205	205	205	205	205	205	205	205.000000	...	205.000000	205	205	205	205.00
unique	NaN	NaN	22	2	2	3	5	3	2	NaN	...	NaN	8	39	37	NaN
top	NaN	NaN	toyota	gas	std	four	sedan	fwd	front	NaN	...	NaN	mpfi	3.62	3.40	NaN
freq	NaN	NaN	32	185	168	114	96	120	202	NaN	...	NaN	94	23	20	NaN
mean	0.834146	122.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	98.756585	...	126.907317	NaN	NaN	NaN	10.142
std	1.245307	31.681008	NaN	NaN	NaN	NaN	NaN	NaN	NaN	6.021776	...	41.642693	NaN	NaN	NaN	3.9720
min	-2.000000	65.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	86.600000	...	61.000000	NaN	NaN	NaN	7.0000
25%	0.000000	101.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	94.500000	...	97.000000	NaN	NaN	NaN	8.6000
50%	1.000000	122.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	97.000000	...	120.000000	NaN	NaN	NaN	9.0000
75%	2.000000	137.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	102.400000	...	141.000000	NaN	NaN	NaN	9.4000
max	3.000000	256.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	120.900000	...	326.000000	NaN	NaN	NaN	23.000

11 rows × 26 columns

In [78]:

```
#Checking the outlier of horsepower
automobile.loc[automobile['horsepower'] > 288]
```

Out[78]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression ratio
130	0	122	renault	gas	std	four	wagon	fwd	front	96.1	...	132	mpfi	3.46	3.90	8.7
131	2	122	renault	gas	std	two	hatchback	fwd	front	96.1	...	132	mpfi	3.46	3.90	8.7

2 rows × 26 columns

In [79]:

```
#Excluding the outlier data for horsepower
automobile[np.abs(automobile.horsepower-automobile.horsepower.mean())<=(3*automobile.horsep
ower.std())]
```

Out[79]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compre ratio
0	3	122	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.00
1	3	122	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.00
2	1	122	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.00
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.00
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.00
5	2	122	audi	gas	std	two	sedan	fwd	front	99.8	...	136	mpfi	3.19	3.40	8.50
6	1	158	audi	gas	std	four	sedan	fwd	front	105.8	...	136	mpfi	3.19	3.40	8.50
7	1	122	audi	gas	std	four	wagon	fwd	front	105.8	...	136	mpfi	3.19	3.40	8.50
8	1	158	audi	gas	turbo	four	sedan	fwd	front	105.8	...	131	mpfi	3.13	3.40	8.30
9	0	122	audi	gas	turbo	two	hatchback	4wd	front	99.5	...	131	mpfi	3.13	3.40	7.00
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
195	-1	74	volvo	gas	std	four	wagon	rwd	front	104.3	...	141	mpfi	3.78	3.15	9.50
196	-2	103	volvo	gas	std	four	sedan	rwd	front	104.3	...	141	mpfi	3.78	3.15	9.50
197	-1	74	volvo	gas	std	four	wagon	rwd	front	104.3	...	141	mpfi	3.78	3.15	9.50
198	-2	103	volvo	gas	turbo	four	sedan	rwd	front	104.3	...	130	mpfi	3.62	3.15	7.50
199	-1	74	volvo	gas	turbo	four	wagon	rwd	front	104.3	...	130	mpfi	3.62	3.15	7.50
200	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.50
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	8.70
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	173	mpfi	3.58	2.87	8.80
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1	...	145	idi	3.01	3.40	23.00
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.50

203 rows × 26 columns

In [80]:

```
# Find out the number of invalid value
automobile['bore'].loc[automobile['bore'] == '?']
```

Out[80]:

```
55    ?
56    ?
57    ?
58    ?
Name: bore, dtype: object
```

In [81]:

```
# Replace the non-numeric value to null and conver the datatype
automobile['bore'] = pd.to_numeric(automobile['bore'],errors='coerce')
automobile.dtypes
```

Out[81]:

```
symboling              int64
normalized-losses      int32
make                  object
fuel-type             object
aspiration            object
num-of-doors          object
body-style            object
drive-wheels          object
engine-location       object
wheel-base           float64
length               float64
width                float64
height               float64
curb-weight            int64
engine-type           object
num-of-cylinders      object
engine-size            int64
fuel-system           object
bore                 float64
stroke                object
compression-ratio    float64
horsepower             int32
peak-rpm              object
city-mpg               int64
highway-mpg            int64
price                  int32
dtype: object
```

In [82]:

```
# Replace the non-number value to null and convert the datatype
automobile['stroke'] = pd.to_numeric(automobile['stroke'],errors='coerce')
automobile.dtypes
```

Out[82]:

```
symboling              int64
normalized-losses      int32
make                  object
fuel-type             object
aspiration            object
num-of-doors          object
body-style            object
drive-wheels          object
engine-location       object
```

```
wheel-base          float64
length              float64
width               float64
height              float64
curb-weight           int64
engine-type          object
num-of-cylinders     object
engine-size           int64
fuel-system          object
bore                float64
stroke              float64
compression-ratio   float64
horsepower            int32
peak-rpm             object
city-mpg              int64
highway-mpg           int64
price                 int32
dtype: object
```

In [83]:

```
# Convert the non-numeric data to null and convert the datatype
automobile['peak-rpm'] = pd.to_numeric(automobile['peak-rpm'],errors='coerce')
automobile.dtypes
```

Out[83]:

```
symboling             int64
normalized-losses     int32
make                 object
fuel-type            object
aspiration           object
num-of-doors         object
body-style           object
drive-wheels         object
engine-location      object
wheel-base          float64
length              float64
width               float64
height              float64
curb-weight           int64
engine-type          object
num-of-cylinders     object
engine-size           int64
fuel-system          object
bore                float64
```

```
stroke              float64
compression-ratio    float64
horsepower           int32
peak-rpm            float64
city-mpg             int64
highway-mpg          int64
price               int32
dtype: object
```

In [84]:

```
# remove the records which are having the value '?'
automobile['num-of-doors'].loc[automobile['num-of-doors'] == '?']
automobile = automobile[automobile['num-of-doors'] != '?']
automobile['num-of-doors'].loc[automobile['num-of-doors'] == '?']
```

Out[84]:

```
27    ?
63    ?
Name: num-of-doors, dtype: object
```

Out[84]:

```
Series([], Name: num-of-doors, dtype: object)
```

In [85]:

```
automobile['num-of-doors'].loc[automobile['num-of-doors'] == '?']
```

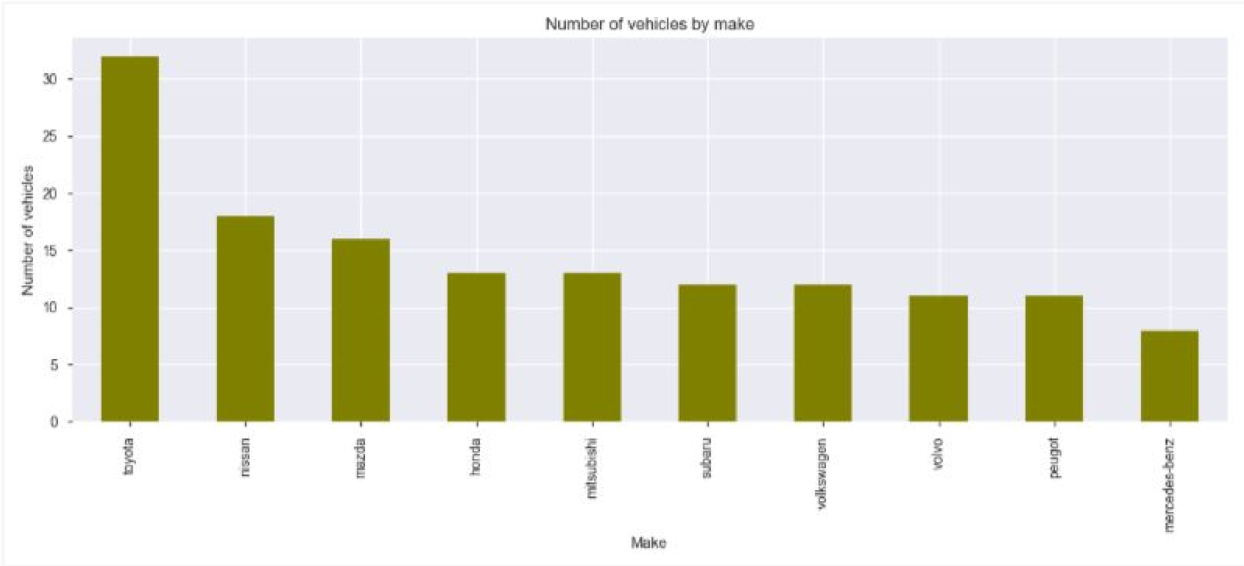
Out[85]:

```
Series([], Name: num-of-doors, dtype: object)
```

Univariate Analysis

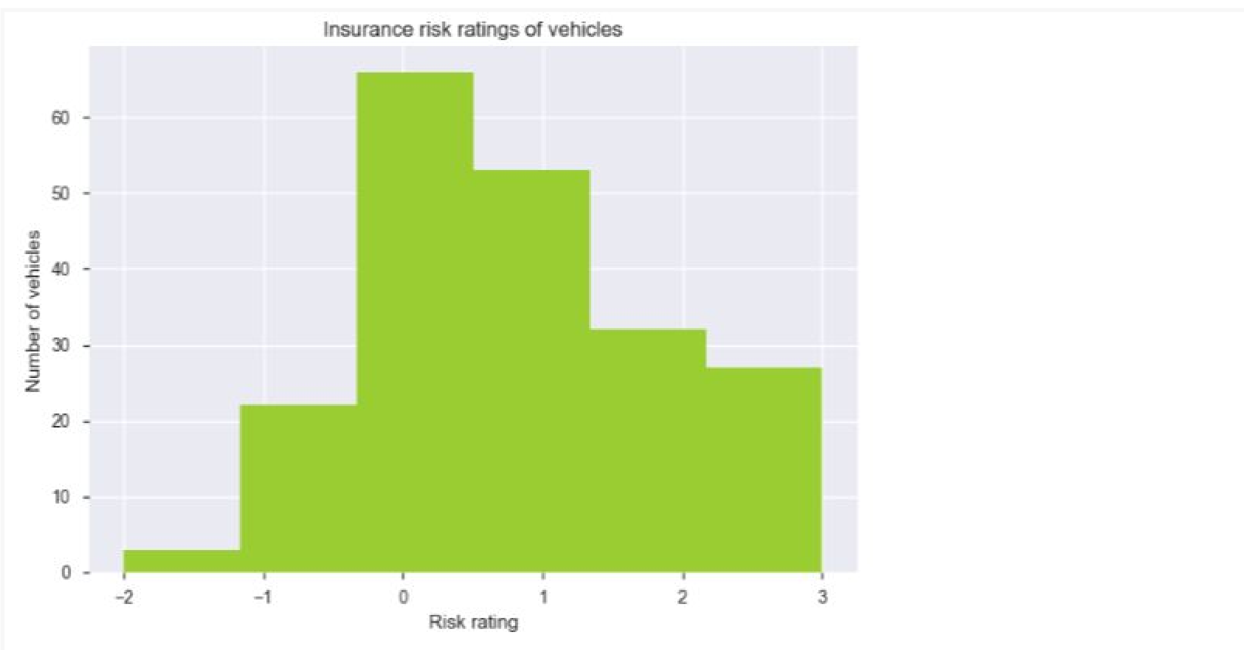
In [86]:

```
#Make wise vehicles manufactured
automobile.make.value_counts().nlargest(10).plot(kind='bar', figsize=(15,5), color='olive')
plt.title("Number of vehicles by make")
plt.ylabel('Number of vehicles')
plt.xlabel('Make');
```



In [87]:

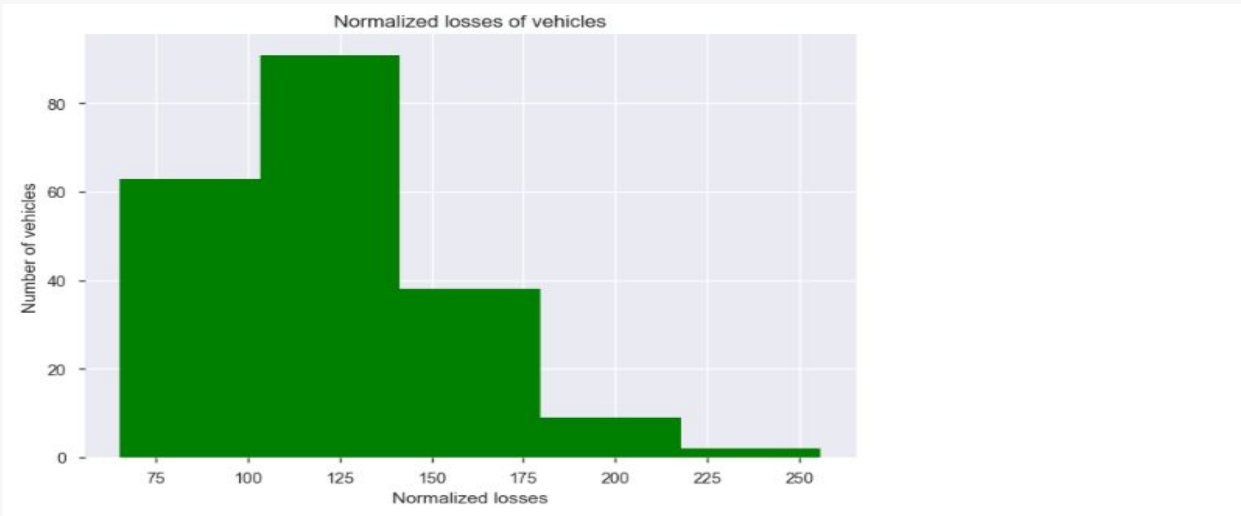
```
#Insurance risk
automobile.symboling.hist(bins=6,color='yellowgreen');
plt.title("Insurance risk ratings of vehicles")
plt.ylabel('Number of vehicles')
plt.xlabel('Risk rating');
```



In [88]:

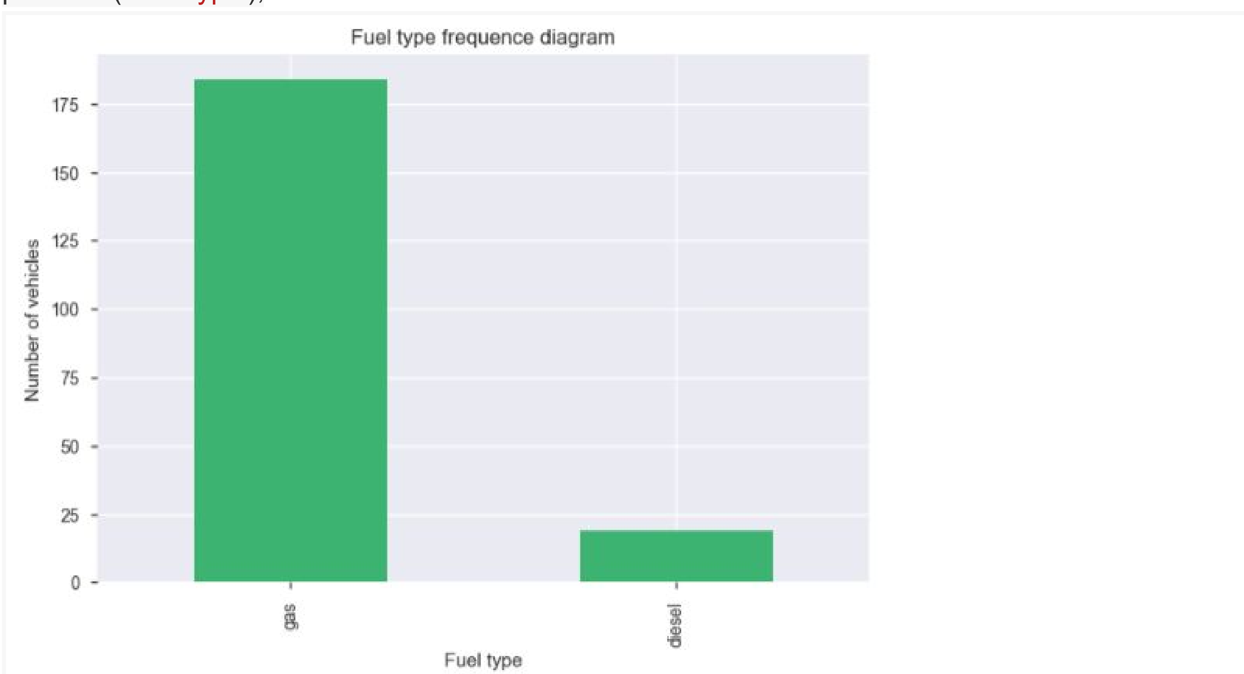
```
#Normalized losses
```

```
automobile['normalized-losses'].hist(bins=5,color='g');
plt.title("Normalized losses of vehicles")
plt.ylabel('Number of vehicles')
plt.xlabel('Normalized losses');
```



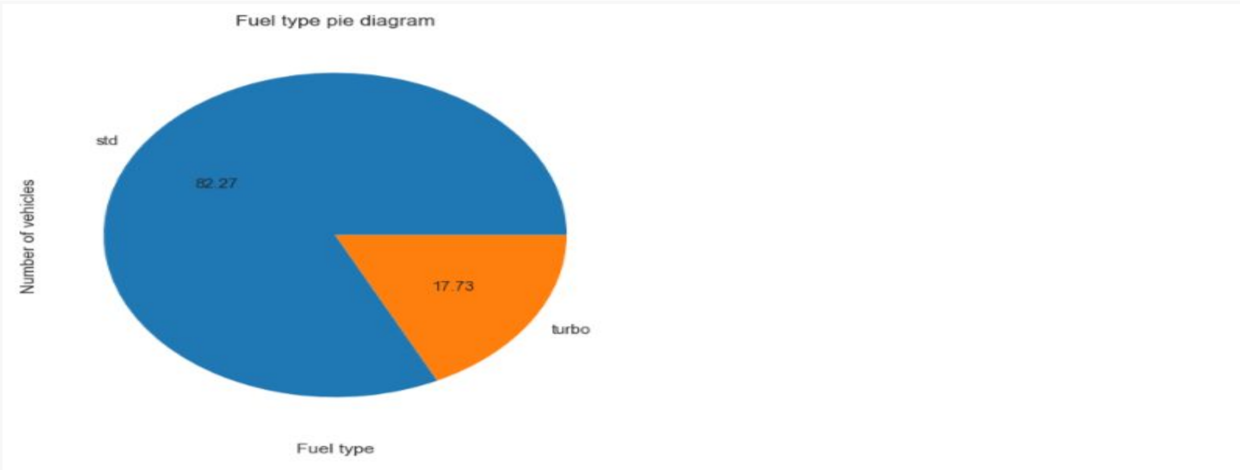
In [89]:

```
#Fuel Type Histogram
automobile['fuel-type'].value_counts().plot(kind='bar',color='mediumseagreen')
plt.title("Fuel type frequence diagram")
plt.ylabel('Number of vehicles')
plt.xlabel('Fuel type');
```



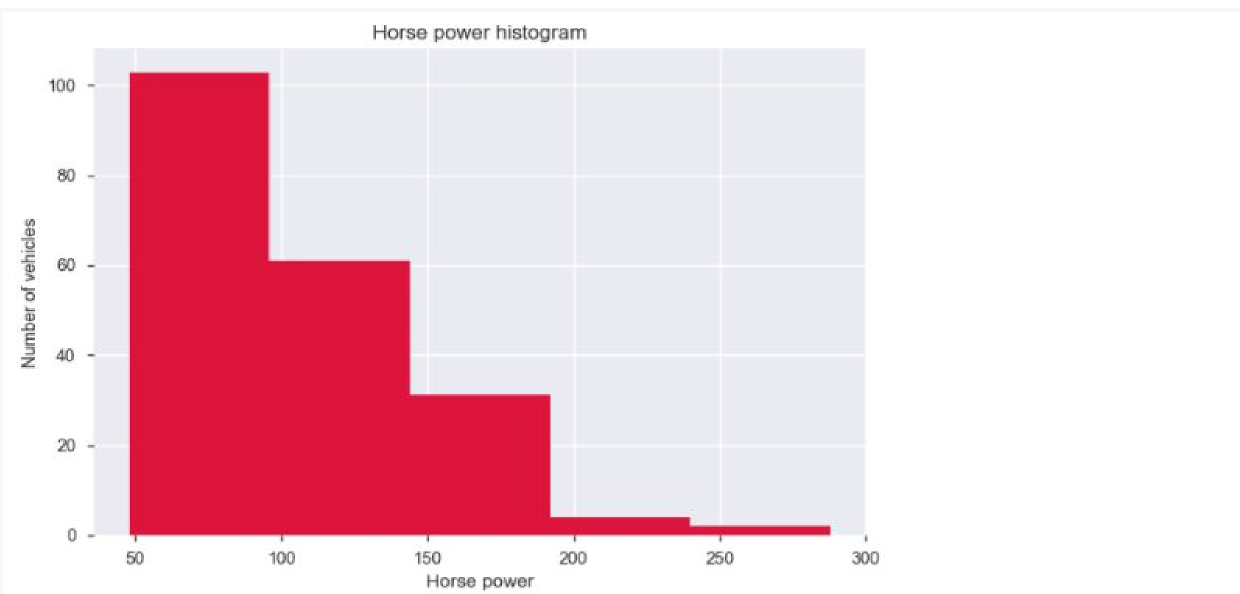
In [90]:

```
#Fuel type PIE chart
automobile['aspiration'].value_counts().plot.pie(figsize=(6,6), autopct='%.2f')
plt.title("Fuel type pie diagram")
plt.ylabel('Number of vehicles')
plt.xlabel('Fuel type');
```



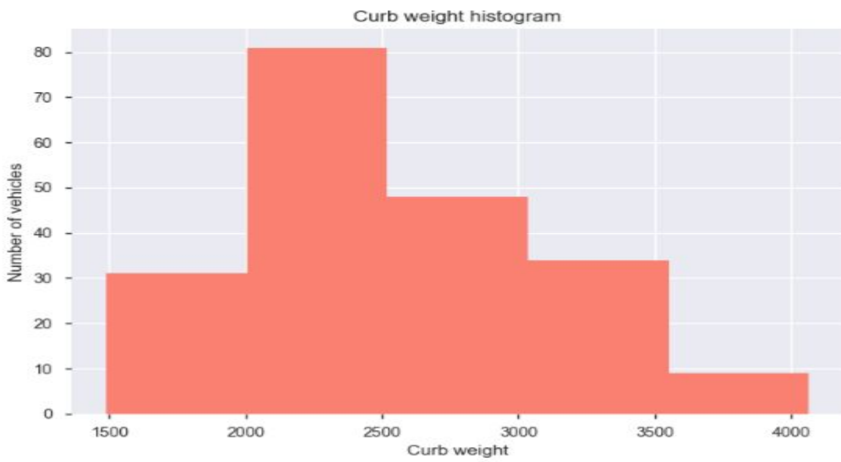
In [91]:

```
#Horse Power histogram
automobile.horsepower[np.abs(automobile.horsepower-automobile.horsepower.mean())<=(3*autom
obile.horsepower.std())].hist(bins=5,color='crimson');
plt.title("Horse power histogram")
plt.ylabel('Number of vehicles')
plt.xlabel('Horse power');
```



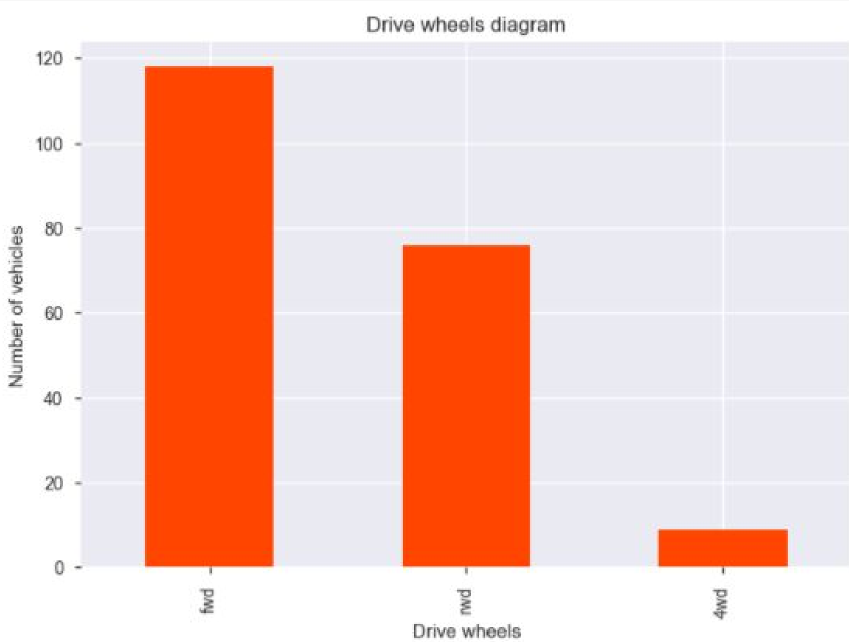
In [92]:

```
#Curb weight histogram
automobile['curb-weight'].hist(bins=5,color='salmon');
plt.title("Curb weight histogram")
plt.ylabel('Number of vehicles')
plt.xlabel('Curb weight');
```

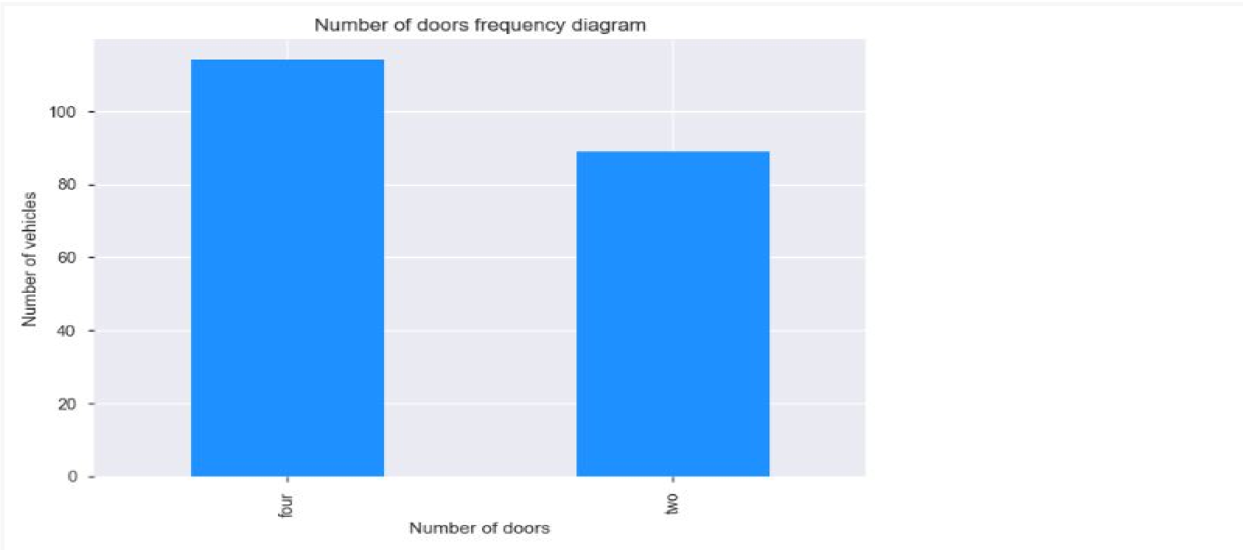


In [93]:

```
#Drive Wheels bar chart
automobile['drive-wheels'].value_counts().plot(kind='bar',color='orangered')
plt.title("Drive wheels diagram")
plt.ylabel('Number of vehicles')
plt.xlabel('Drive wheels');
```



In [94]:
```
#Number of doors bar chart
automobile['num-of-doors'].value_counts().plot(kind='bar',color='dodgerblue')
plt.title("Number of doors frequency diagram")
plt.ylabel('Number of vehicles')
plt.xlabel('Number of doors');
```

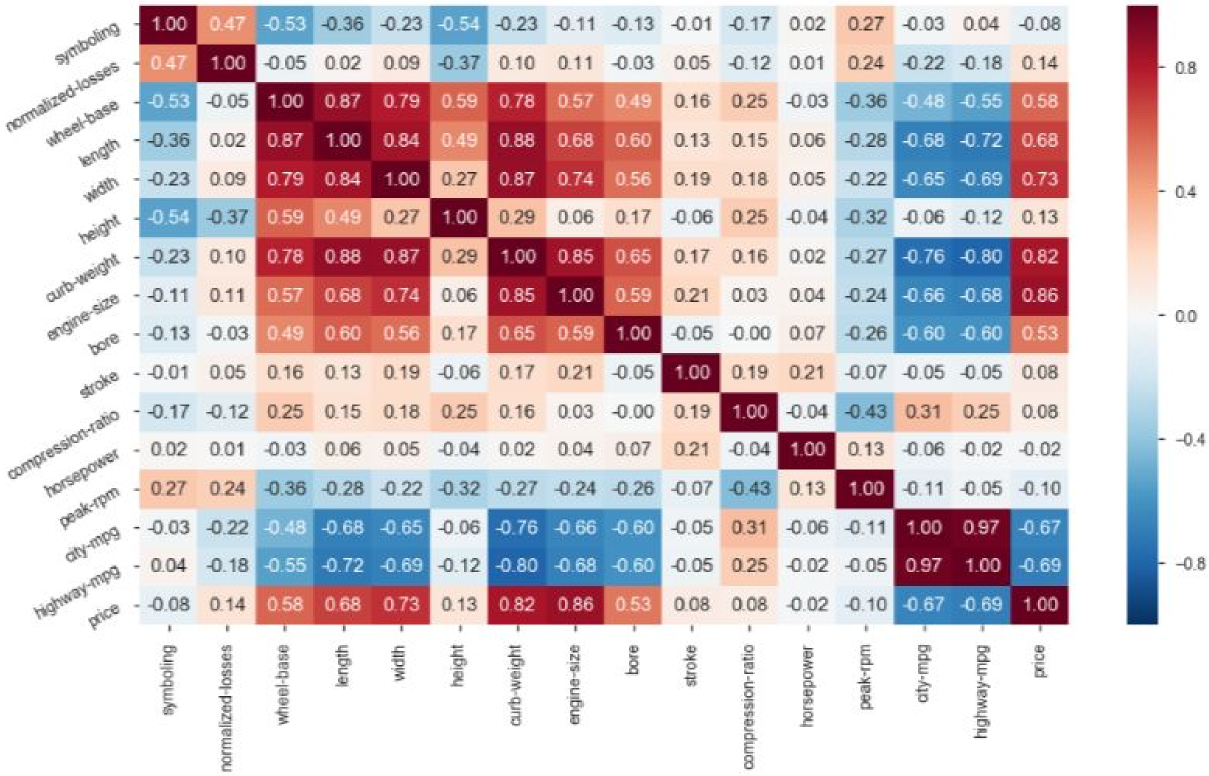


Make, Curb-weight, Drive wheels Analysis

Make : Toyota car has most number of vehicles with more than 40% than the 2nd highest Nissan
Curb-weight : Cars Curb weight between 1500 and 4000 approximately
Drive wheels : front wheel drive has most number of cars followed by rear wheel and four wheel. There are very less number of cars for four wheel drive.

In [95]:
```
corr = automobile.corr()
sns.set_context("notebook", font_scale=1.0, rc={"lines.linewidth": 2.5})
plt.figure(figsize=(13,7))
a = sns.heatmap(corr, annot=True, fmt='.2f')
rotx = a.set_xticklabels(a.get_xticklabels(), rotation=90)
roty = a.set_yticklabels(a.get_yticklabels(), rotation=30)
```

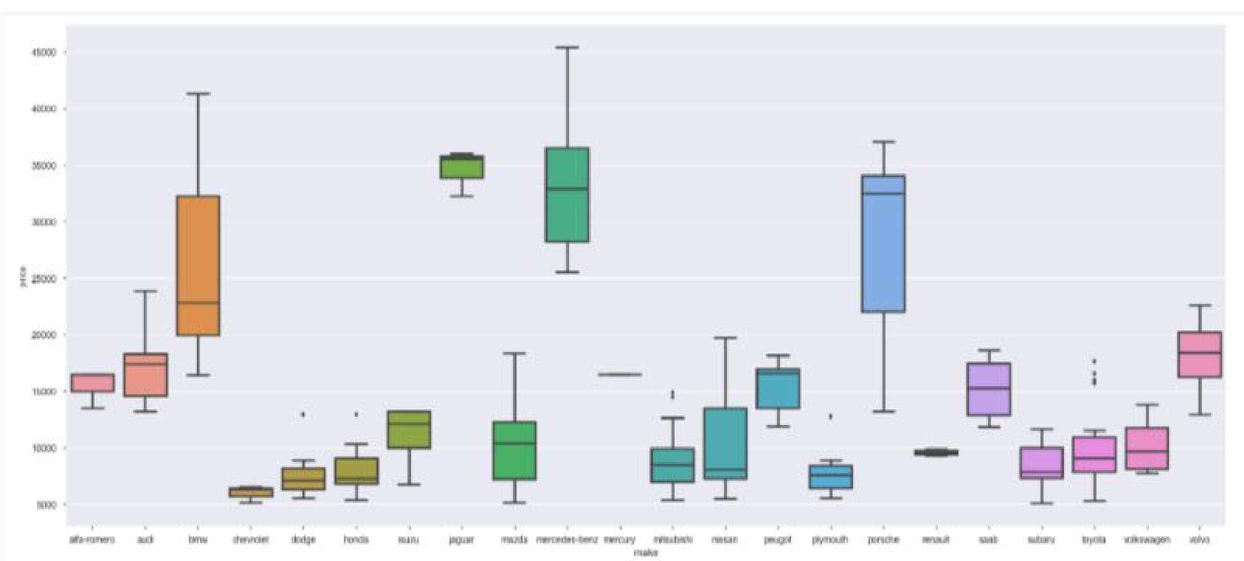
Out[95]:
```
<matplotlib.figure.Figure at 0x38436959e8>
```



Bivariate Analysis

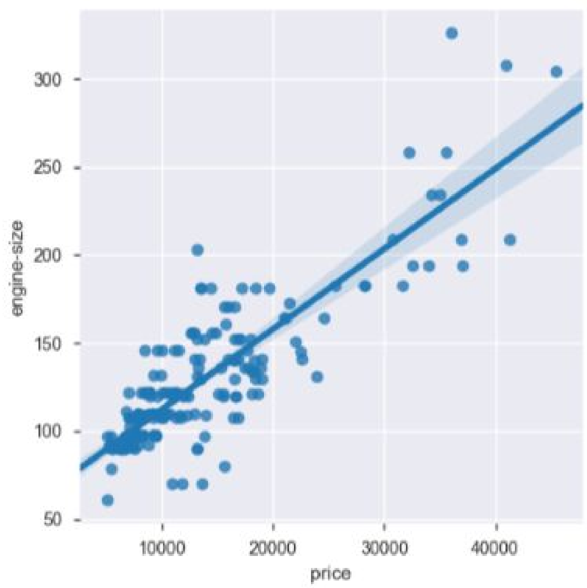
In [96]:

```
#Boxplot of Price and make
plt.rcParams['figure.figsize']=(23,10)
ax = sns.boxplot(x="make", y="price", data=automobile)
```



In [97]:

```
#Scatter plot of price and engine size
g = sns.lmplot('price',"engine-size", automobile);
```

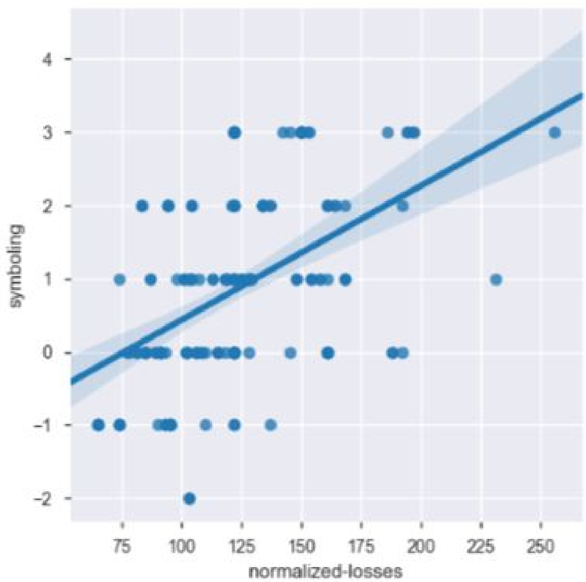


Variable analysis to see its impact on automobile pricing

Findings: Below are our findings on the make and price of the car
The most expensive car is manufacture by Mercedes benz and the least expensive is Chevrolet
The premium cars costing more than 20000 are BMW, Jaquar, Mercedes benz and Porsche
Less expensive cars costing less than 10000 are Chevrolet, Dodge, Honda, Mitsubishi, Plymoth and Subaru
Rest of the cars are in the midrange between 10000 and 20000 which has the highest number of cars

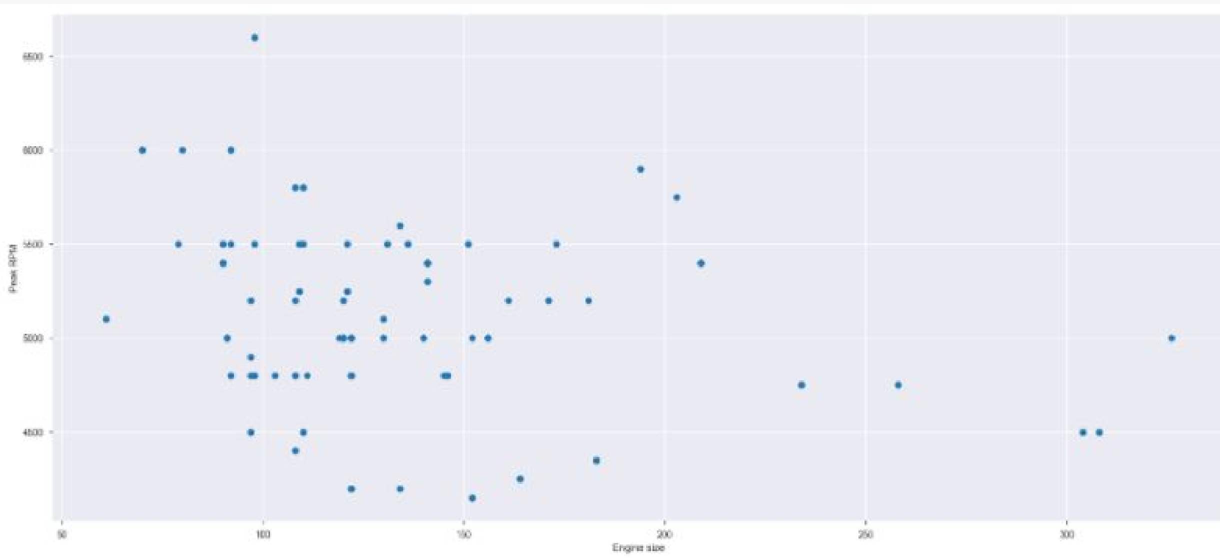
In [98]:

```
#Scatter plot of normalized losses and symboling
g = sns.lmplot('normalized-losses',"symboling", automobile);
```



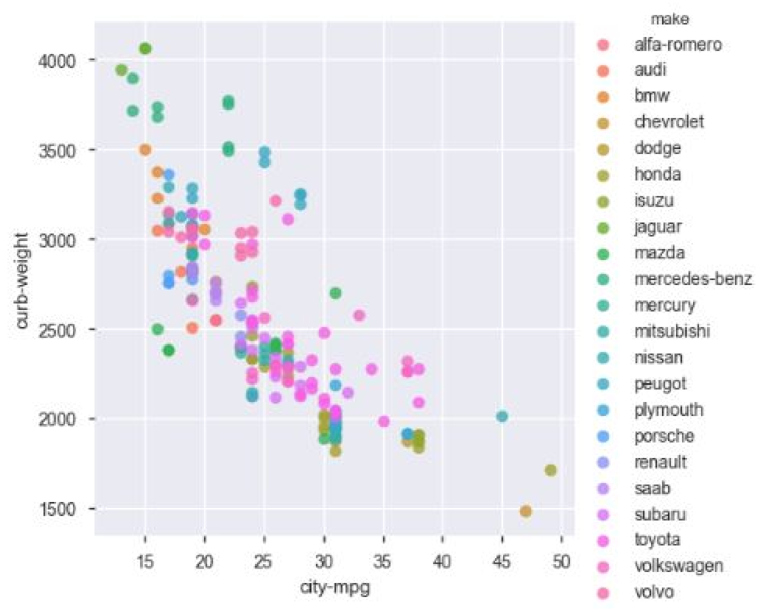
In [99]:

```
#Scatter plot of Engine size and Peak RPM
plt.scatter(automobile['engine-size'],automobile['peak-rpm'])
plt.xlabel('Engine size')
plt.ylabel('Peak RPM');
```



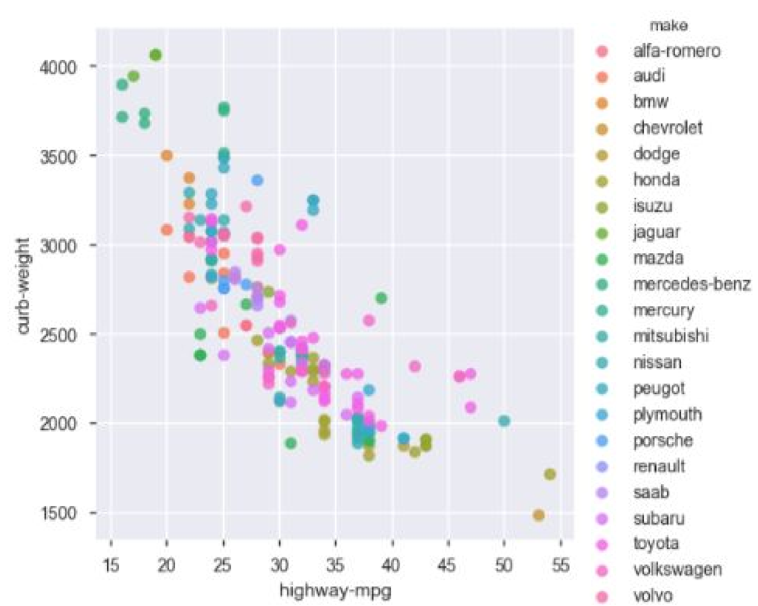
In [100]:

```
#Scatter plot of City and Highway MPG, Curb weight based on Make of the car
g = sns.lmplot('city-mpg',"curb-weight", automobile, hue="make", fit_reg=False);
```



In [101]:

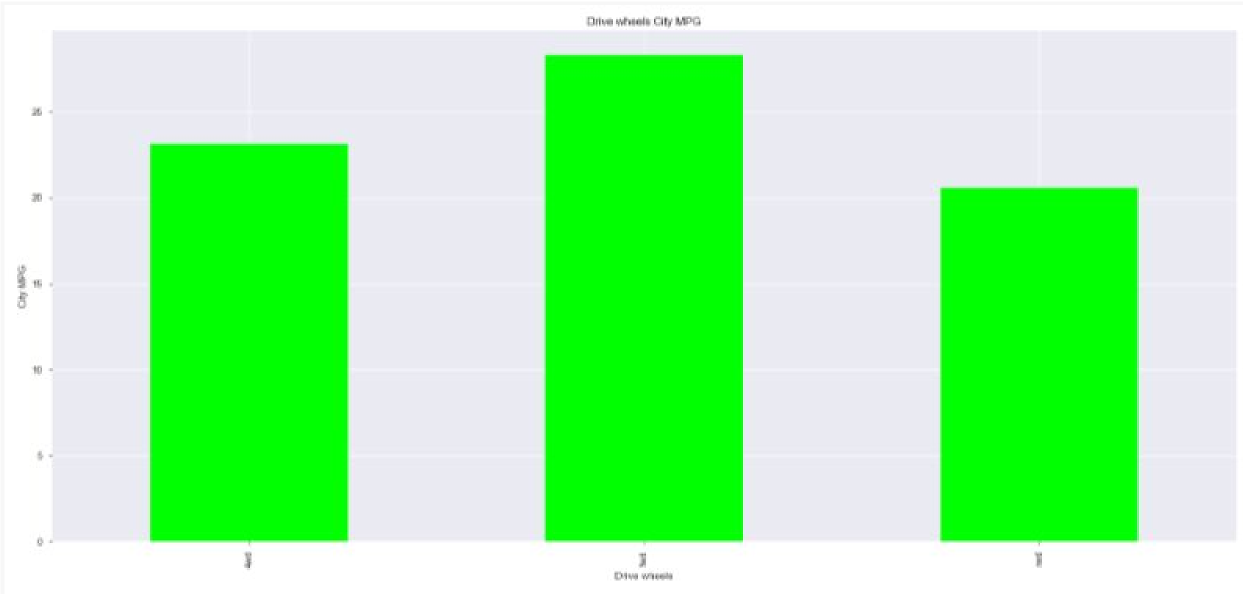
```
g = sns.lmplot('highway-mpg',"curb-weight", automobile, hue="make",fit_reg=False);
```



In [102]:

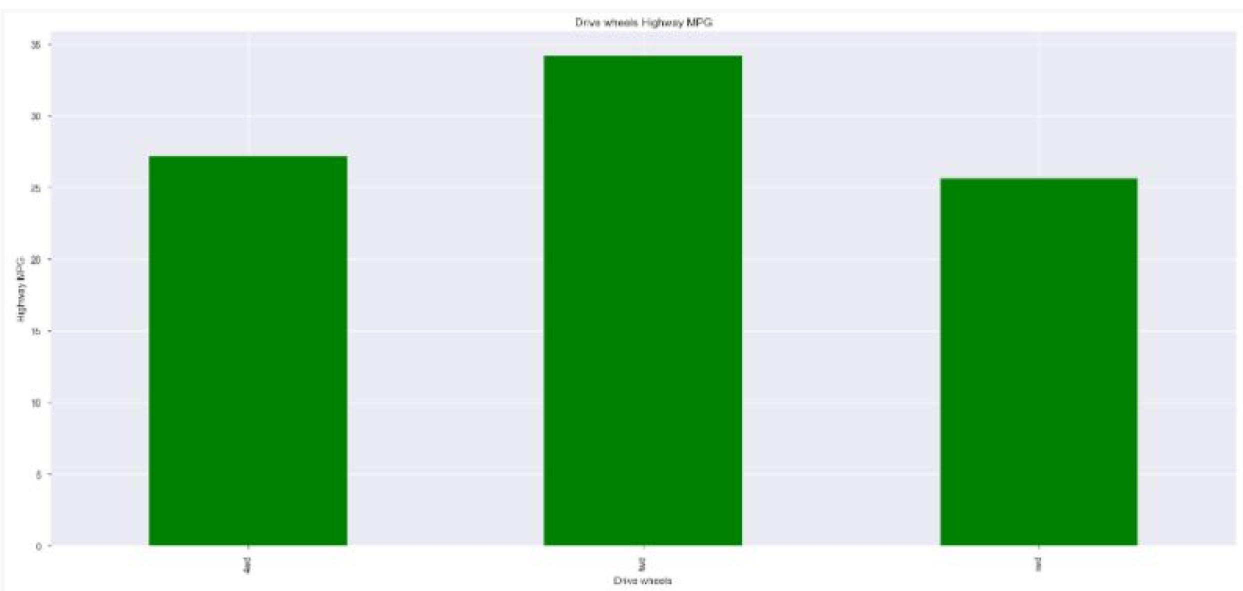
```
#Drive wheels and City MPG bar chart
automobile.groupby('drive-wheels')['city-mpg'].mean().plot(kind='bar', color = 'lime');
plt.title("Drive wheels City MPG")
```

```
plt.ylabel('City MPG')
plt.xlabel('Drive wheels');
```



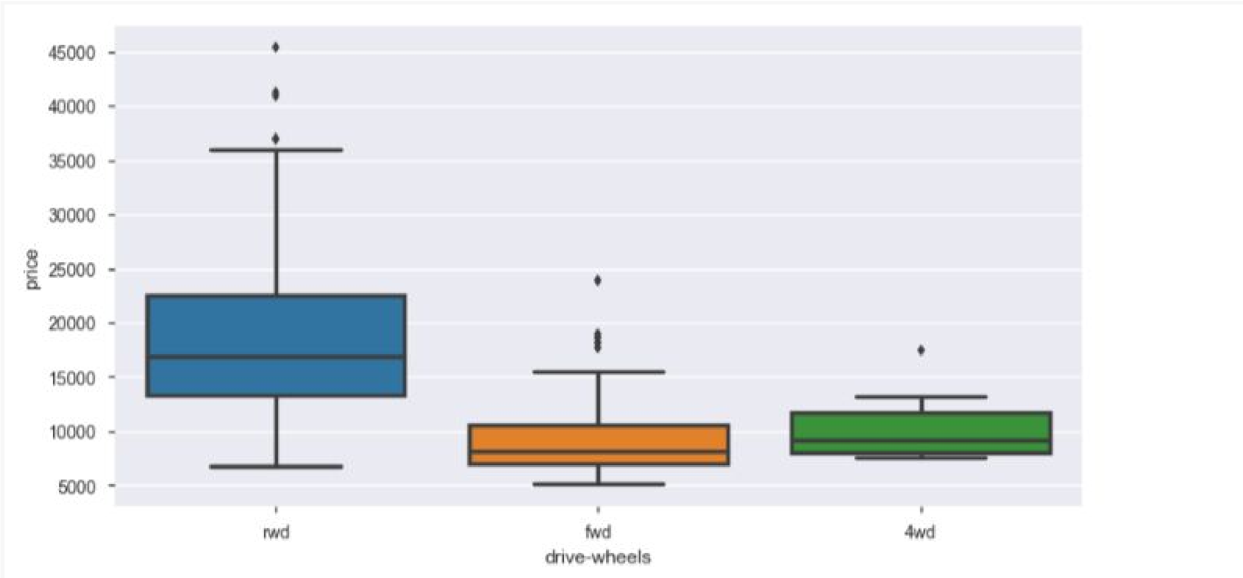
In [103]:

```
#Drive wheels and Highway MPG bar chart
automobile.groupby('drive-wheels')['highway-mpg'].mean().plot(kind='bar', color = 'green');
plt.title("Drive wheels Highway MPG")
plt.ylabel('Highway MPG')
plt.xlabel('Drive wheels');
```



In [104]:

```
#Boxplot of Drive wheels and Price
plt.rcParams['figure.figsize']=(10,5)
ax = sns.boxplot(x="drive-wheels", y="price", data=automobile)
```



In [105]:

```
#Normalized losses based on body style and no. of doors
pd.pivot_table(automobile,index=['body-style','num-of-doors'],
values='normalized-losses').plot(kind='bar',color='orange')
plt.title("Normalized losses based on body style and no. of doors")
plt.ylabel('Normalized losses')
plt.xlabel('Body style and No. of doors');
```