In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

Uploading dataset

In [2]:
```
auto_price=pd.read_csv("Automobile_price_data__Raw_.csv")
```

In [3]:
```
# 1st five elements of dataset
auto_price.head()
```

Out[3]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4

5 rows × 26 columns

In [4]:
```
#shape of dataset
auto_price.shape
```

Out[4]:
```
(205, 26)
```

In [5]:
```
#data types of dataset
auto_price.dtypes
```

Out[5]:
```
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 [6]:
```
# No of rows with null values
auto_price.isnull().sum()
```

Out[6]:
```
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 [7]:
```
# Convert object data types to numerical datatype
cols=['bore','stroke','horsepower','peak-rpm','price']
auto_price[cols]=auto_price[cols].apply(pd.to_numeric,args=("coerce",))
```

In [8]:
```
auto_price.isnull().sum()
```

Out[8]:
```
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                 4
stroke               4
compression-ratio    0
horsepower           2
peak-rpm             2
city-mpg             0
highway-mpg          0
price                4
dtype: int64
```

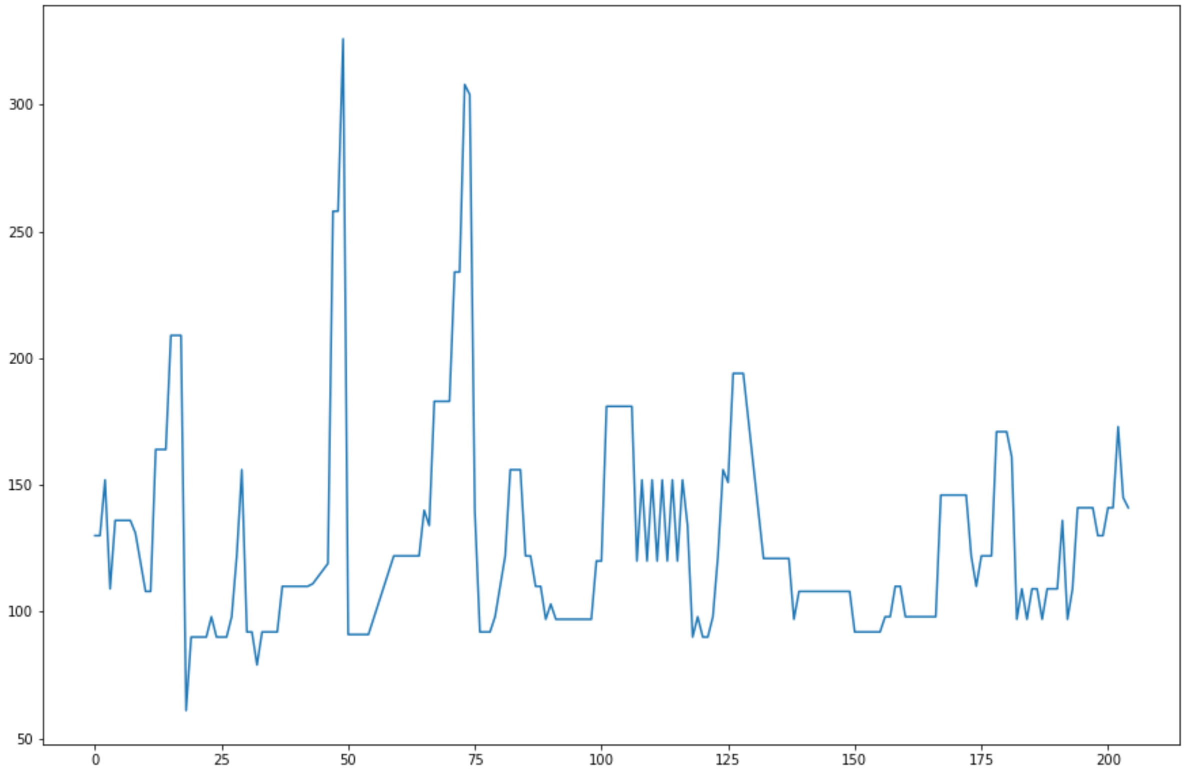
In [9]:
```
auto_price.dropna(inplace=True)
```

Line Plot

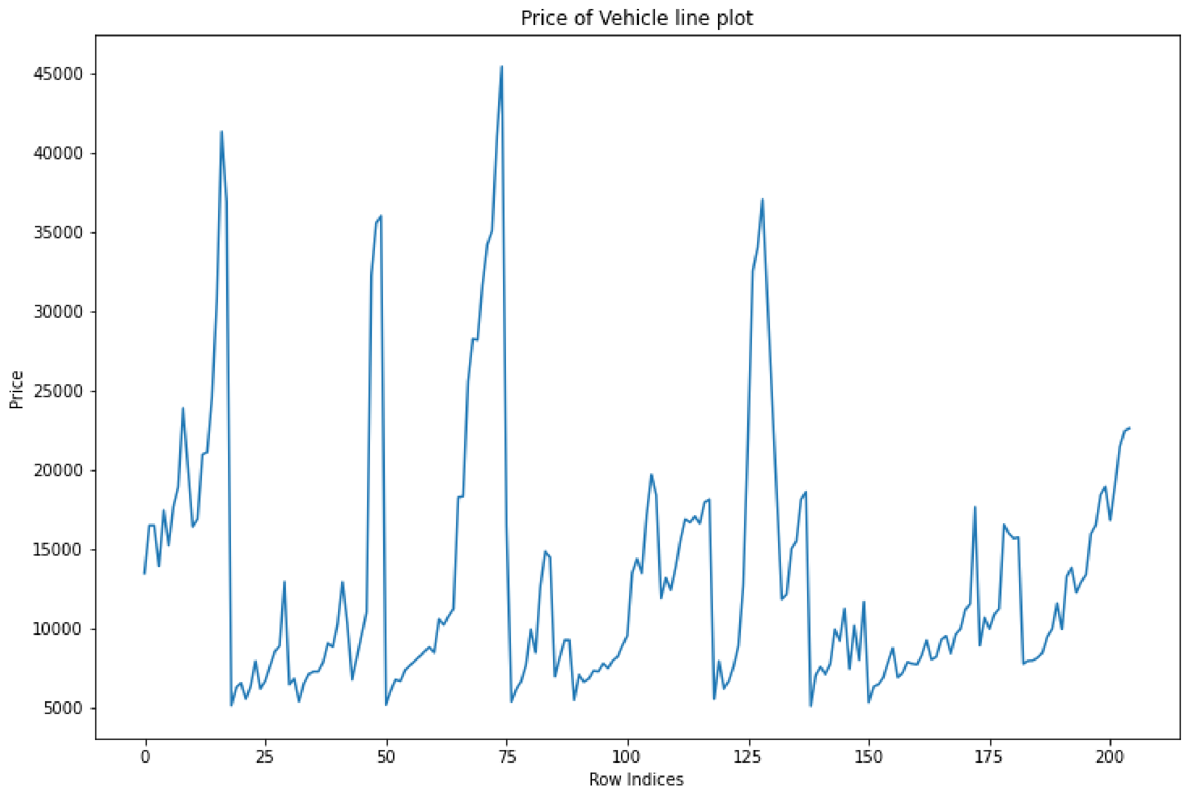
- Relationship Plot (Bivariate)
- Trends - Univariate

In [10]:
```
#Plot function is used to declare which kind of graph we need - It is pandas funct
auto_price["engine-size"].plot(kind="line",figsize=(15,10))
```

Out[10]:
```
<AxesSubplot:>
```



In [11]:
```
sns.set_style=("whitegrid")
plt.figure(figsize=(12,8))
plt.plot(auto_price["price"])
plt.title("Price of Vehicle line plot",size=12)
plt.xlabel("Row Indices")
plt.ylabel("Price")
plt.show()
```



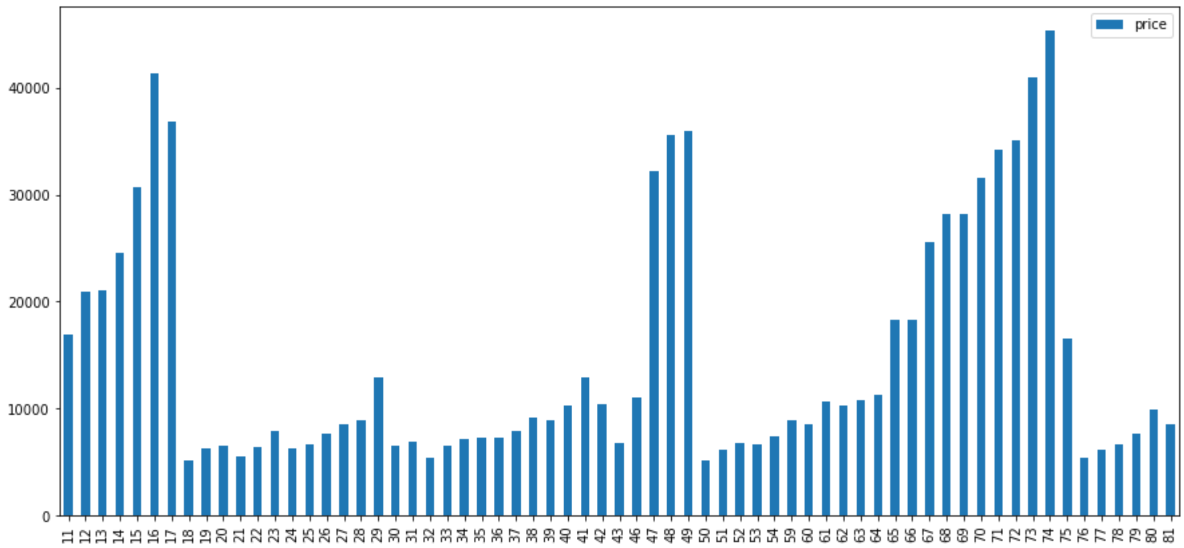
Barplot

- It is used for bivarate analysis
- It is used for cateogorical, nominal , ordinal

- Bar chart used to display value counts of unique values, hieght of bar represents frequency of each catogory

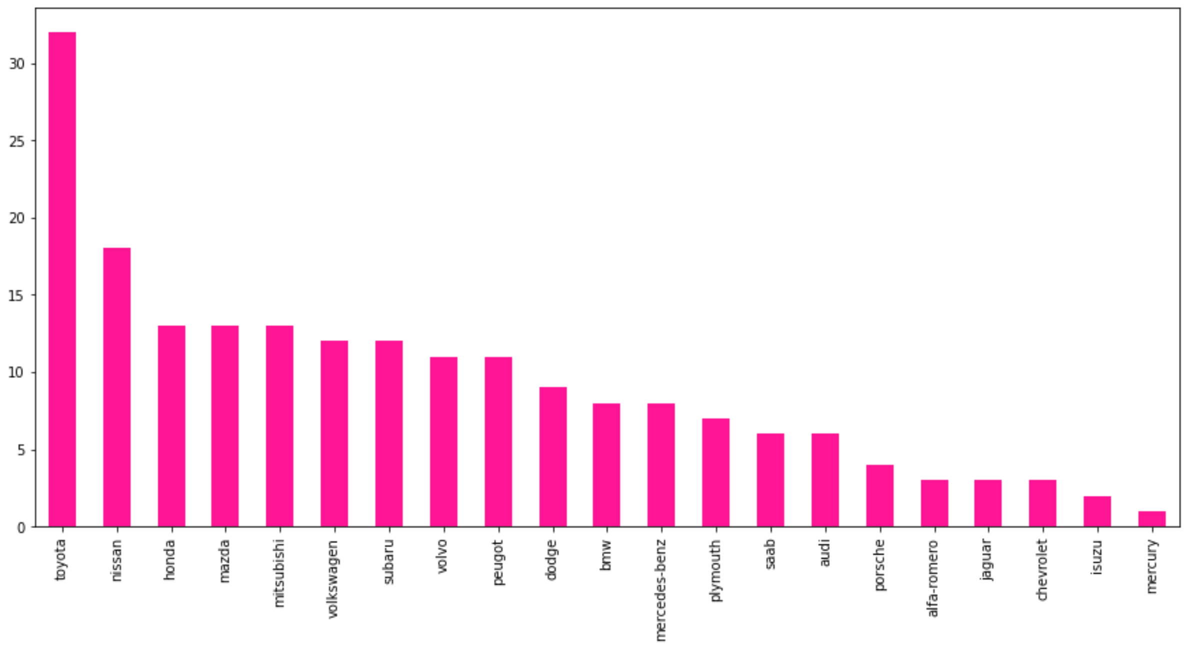
In [12]:
```
sns.set_style=('whitegrid')
auto_price[['price']].iloc[10:75,].plot.bar(figsize = (15,7))
#iloc is used to represent column as starting value as 10 and end with 75
```

Out[12]: <AxesSubplot:>



In [13]:
```
auto_price['make'].value_counts().plot.bar(figsize = (15,7),color="deeppink")
```

Out[13]: <AxesSubplot:>

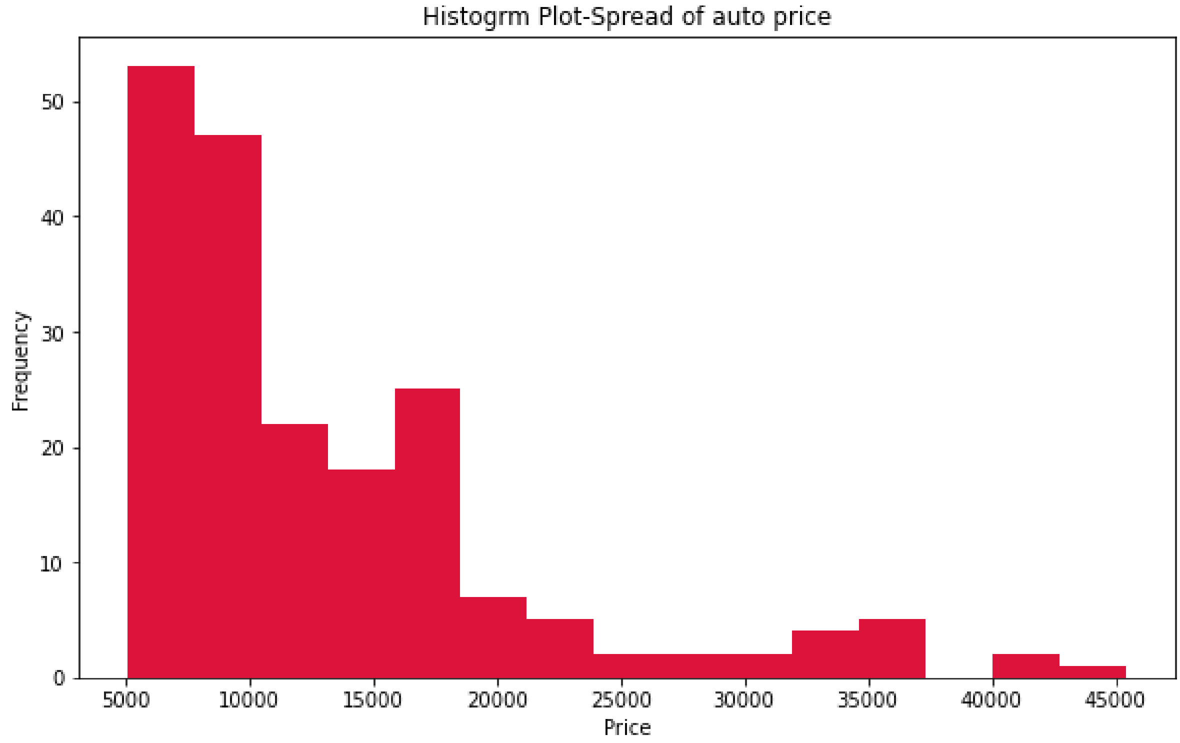


Histogram

- It is created for continious samples- used for study the spread and distribution of data
- It is univarate analysis of data

In [14]:
```
fig=plt.figure(figsize=(10,6))
#bins= no of intervals
plt.hist(auto_price['price'],color="crimson",bins=15)
plt.title("Histogrm Plot-Spread of auto price")
plt.xlabel("Price")
```

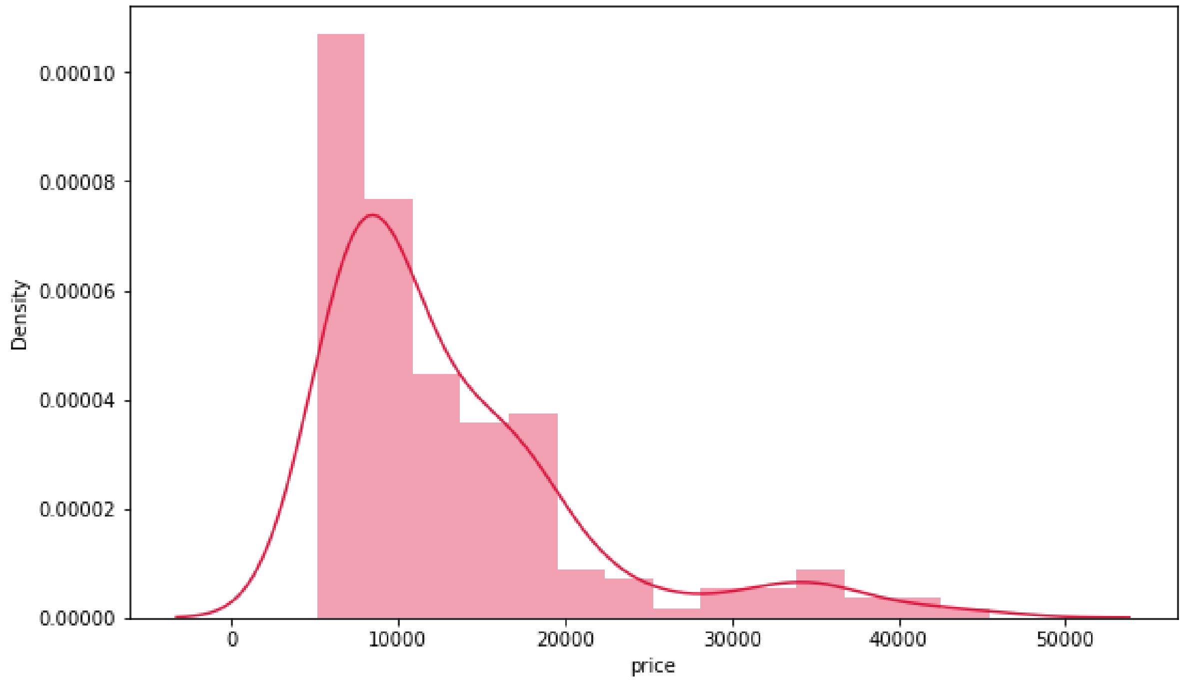
```
plt.ylabel("Frequency")
plt.show()
```



Distribution Plot

- To understand if we have recieved normally distributed data or not, if not then is skewed
- Displays probablity density function
- Continious
- Combination of two plots, line and histogram

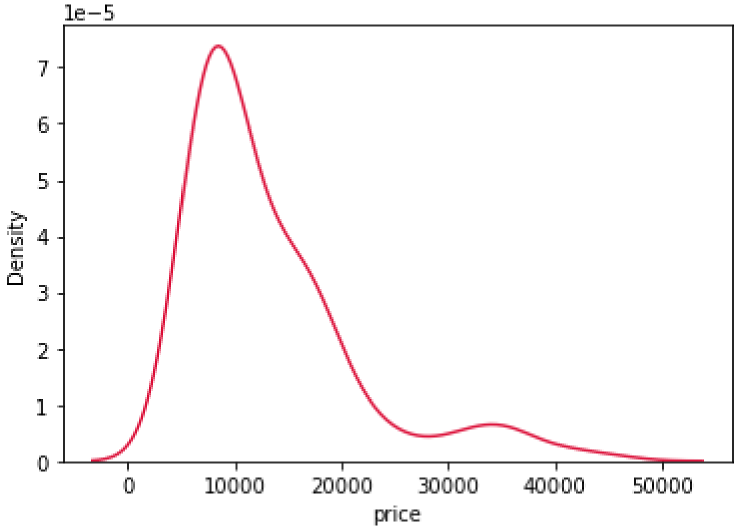
In [15]:
```
fig = plt.figure(figsize = (10,6))
sns.distplot(auto_price['price'], color = 'crimson')
plt.show()
```



In [16]:

```
fig=figsize=(12,8)
sns.distplot(auto_price['price'],color="crimson",hist=False)
#hist = false means= disable histogram plot and show only KDE plot or corner densi
```

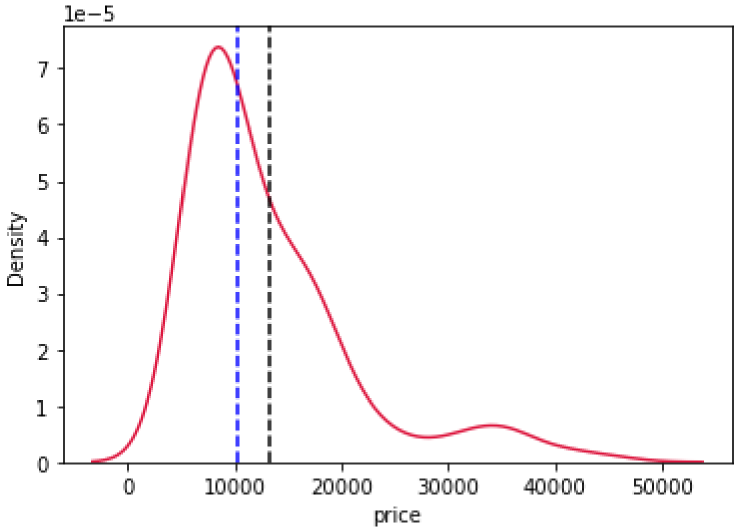
Out[16]: <AxesSubplot:xlabel='price', ylabel='Density'>



In [17]:

```
fig=figsize=(12,8)
sns.distplot(auto_price['price'],color="crimson",hist=False)
#axvline function used for plotting vertical lines
plt.axvline(auto_price['price'].mean(),linestyle="--",color="black")
plt.axvline(auto_price['price'].median(),linestyle="--",color="blue")
```

Out[17]: <matplotlib.lines.Line2D at 0x1f8cb96dac0>



Subplots

- Showing multiple plots combined as one single plot.

In [18]:

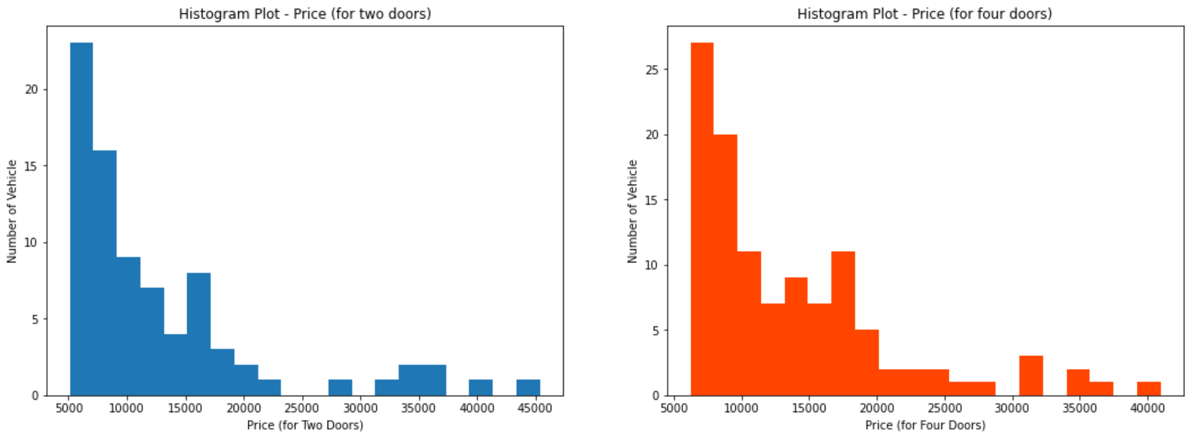
```
plt.figure(figsize = (18,6))
# (1,2,1) => 1st Row, 2nd Col, 1st Index
plt.subplot(1, 2, 1)
plt.hist(auto_price[auto_price['num-of-doors'] == 'two']['price'], bins = 20)
plt.title('Histogram Plot - Price (for two doors)')
plt.xlabel('Price (for Two Doors)')
plt.ylabel('Number of Vehicle')

plt.subplot(1, 2, 2)
```

```
plt.hist(auto_price[auto_price['num-of-doors'] == 'four']['price'], color = 'orang
plt.title('Histogram Plot - Price (for four doors)')
plt.xlabel('Price (for Four Doors)')
plt.ylabel('Number of Vehicle')
```

Out[18]:

```
Text(0, 0.5, 'Number of Vehicle')
```

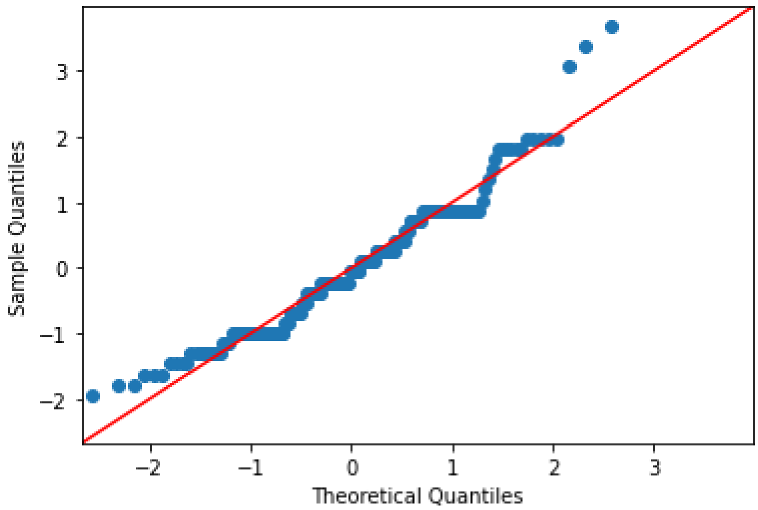


Q-Q Plot

- Test for Normality

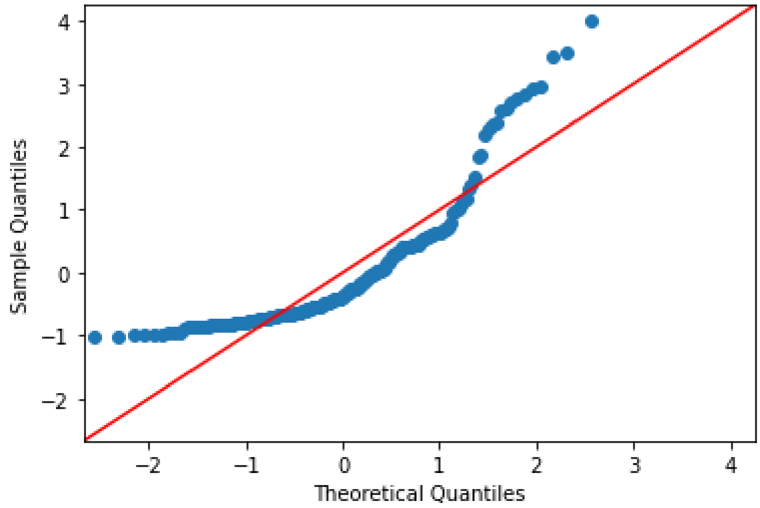
In [19]:

```
import statsmodels.api as stats
#line = to create line at 45 degree, fit- get train
stats.qqplot(auto_price['city-mpg'], line = '45', fit = True)
plt.show()
```



In [20]:

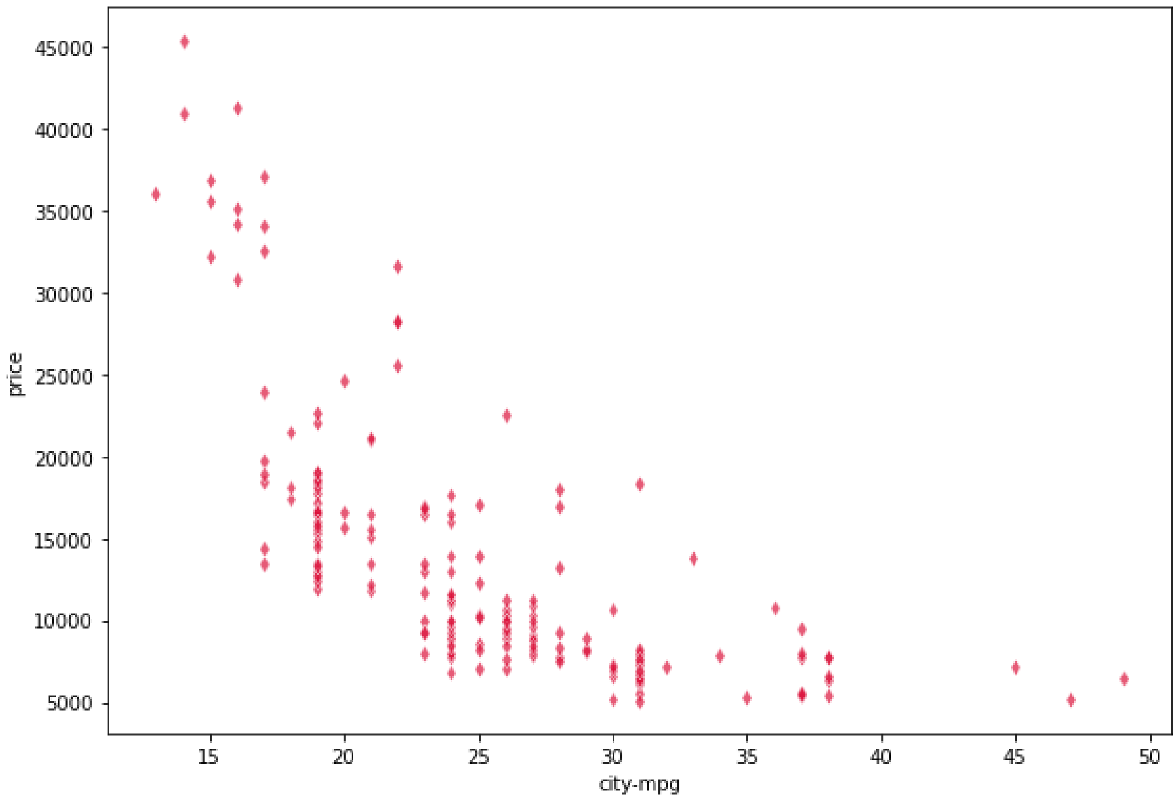
```
import statsmodels.api as stats
stats.qqplot(auto_price['price'],line="45", fit=True)
plt.show()
```



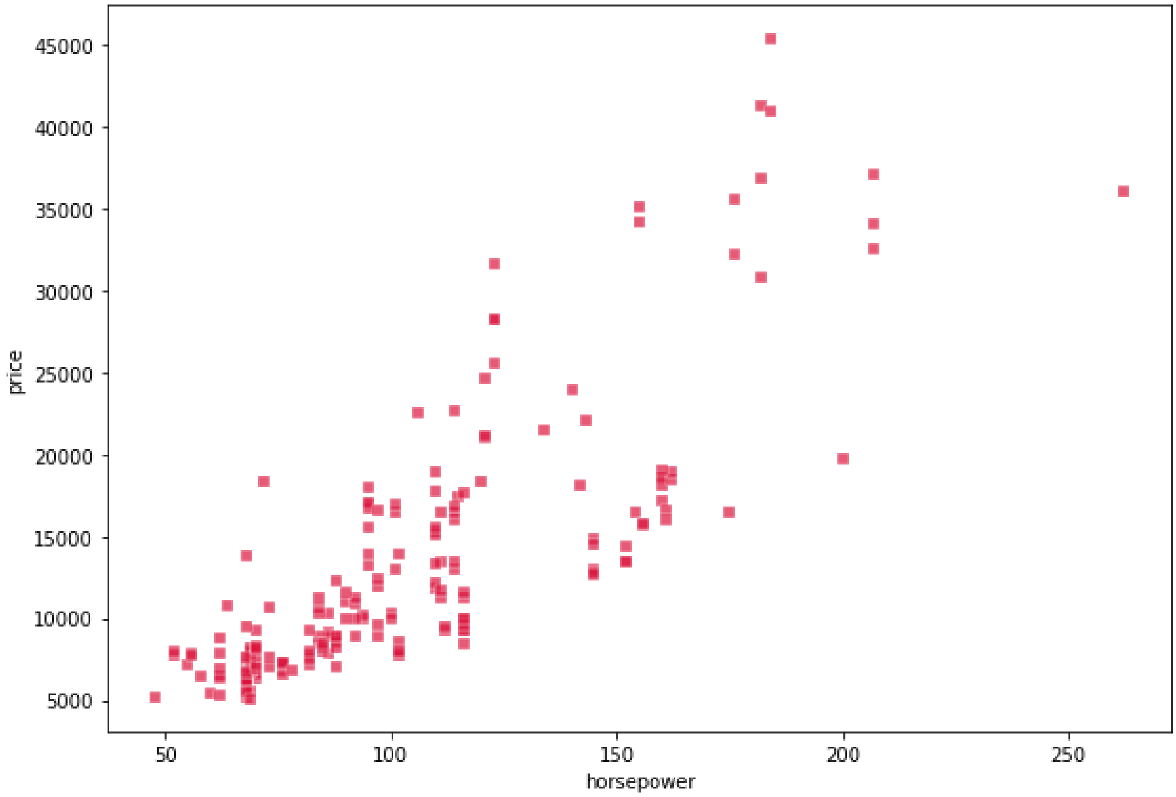
Scatter Plot

- Scatter Plot is plotted between numerical values.
- Shows relationship b/w x & y plots.
- Bivariate

In [21]:
```
plt.figure(figsize=(10,7))
# alpha = transpenrancy
# marker = changing the scatter plots style o,x,X,s,d,>,<,*,.,^
sns.scatterplot(x = auto_price['city-mpg'], y = auto_price['price'], color = 'crim
plt.show()
```



In [22]:
```
plt.figure(figsize=(10,7))
# alpha = transpenrancy
# marker = changing the scatter plots style o,x,X,s,d,>,<,*,.,^
sns.scatterplot(x = auto_price['horsepower'], y = auto_price['price'], color = 'cr
plt.show()
```

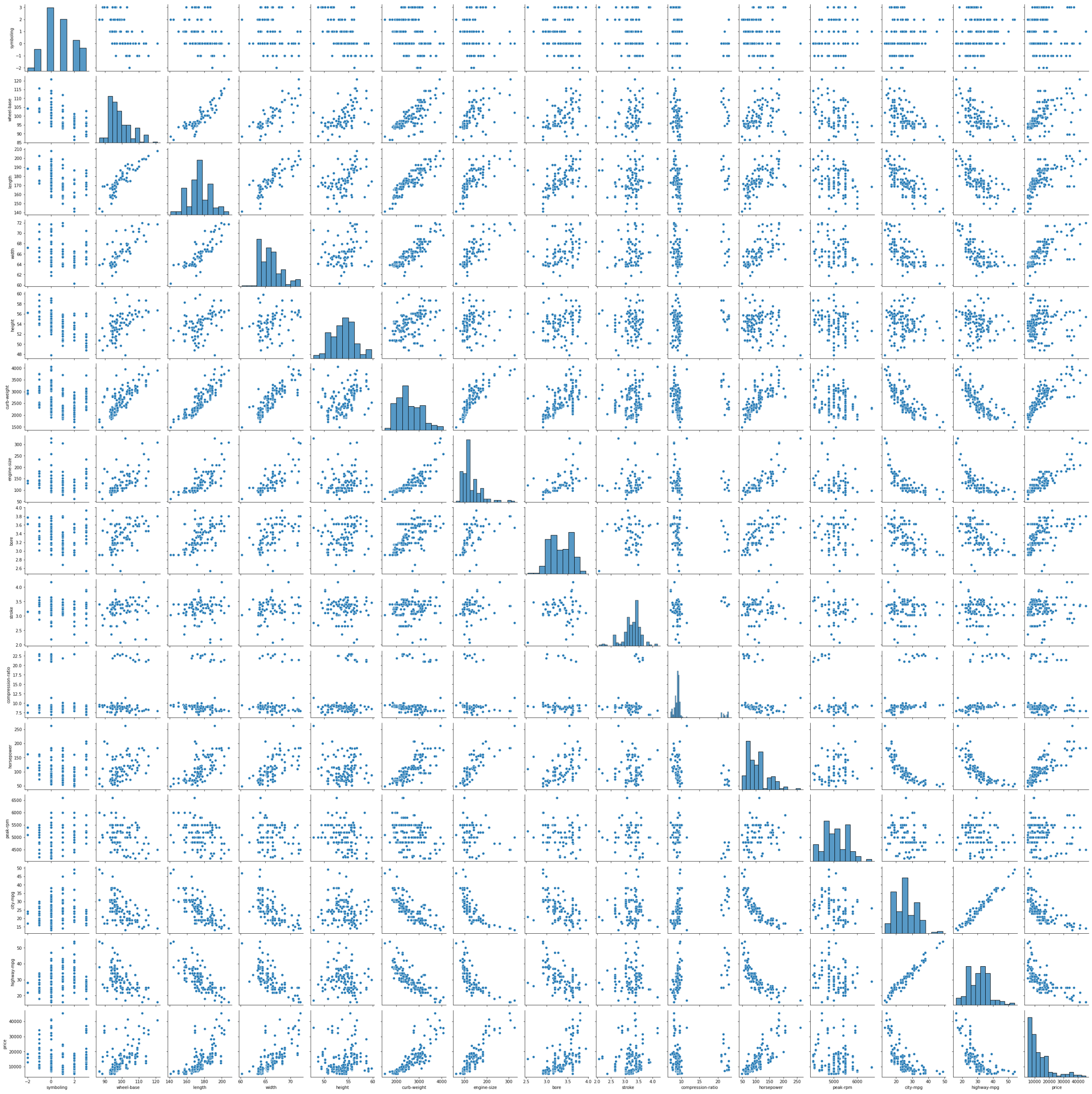


In [23]:

```
sns.pairplot(auto_price)
# it will also show the same result
# diagonally it will show the distribution and non diagonally - scatter plot
```

Out[23]:

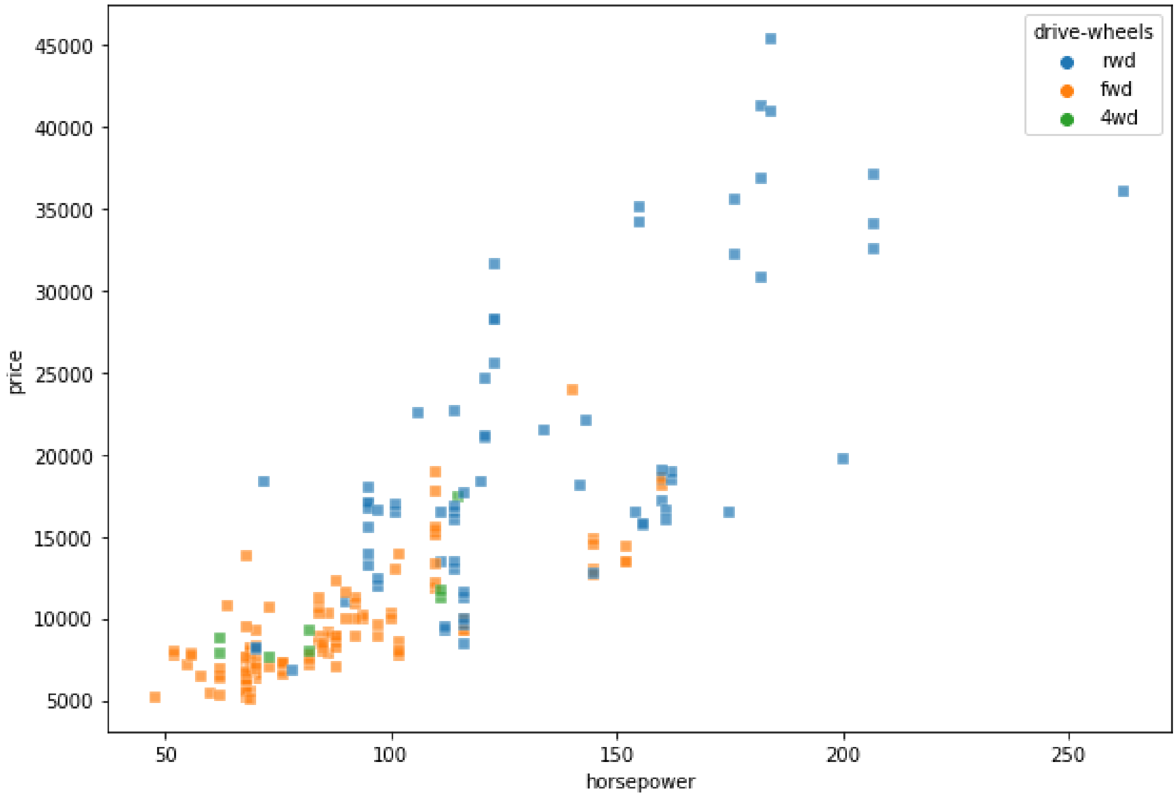
```
<seaborn.axisgrid.PairGrid at 0x1f8ce1de340>
```



In [24]:
```
# hue graph, within scatter plot we can show up the cateogries
# we can increase above size as well by plt.figure=figsize=(25,15)
```

In [25]:
```
plt.figure(figsize  = (10,7))
# alpha = transperancy
# marker = shape of plot, changing the scatter plots style o,x,X,s,d,>,<,*,.,^
sns.scatterplot(x = auto_price['horsepower'], y = auto_price['price'], color = 'cr
                marker = 's', alpha = 0.7)
# hue- autoprice showing here about diffrent cateogries,rwd, fwd or 4wd
```

Out[25]:
```
<AxesSubplot:xlabel='horsepower', ylabel='price'>
```



Heatmap

- Multivariate Data

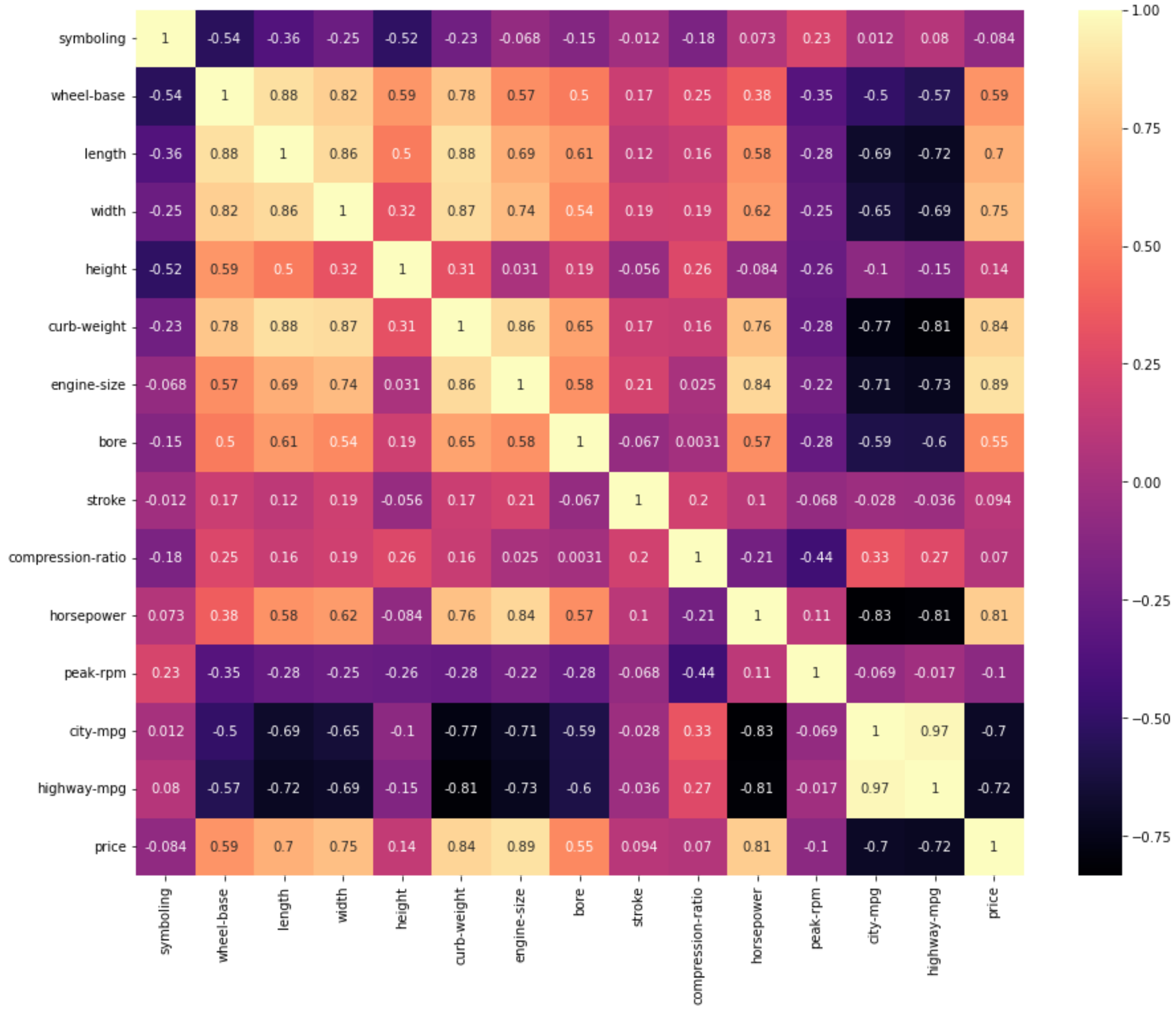
In [26]:
```
auto_price.corr()
```

Out[26]:

	symboling	wheel-base	length	width	height	curb-weight	engine-size	bo
symboling	1.000000	-0.535565	-0.363063	-0.248580	-0.517540	-0.230350	-0.068284	-0.14582
wheel-base	-0.535565	1.000000	0.879222	0.819009	0.592500	0.782720	0.569704	0.49822
length	-0.363063	0.879222	1.000000	0.858084	0.496218	0.881665	0.687479	0.60943
width	-0.248580	0.819009	0.858084	1.000000	0.315834	0.867315	0.740320	0.54431
height	-0.517540	0.592500	0.496218	0.315834	1.000000	0.307732	0.031286	0.18928
curb-weight	-0.230350	0.782720	0.881665	0.867315	0.307732	1.000000	0.857573	0.64580
engine-size	-0.068284	0.569704	0.687479	0.740320	0.031286	0.857573	1.000000	0.58309
bore	-0.145823	0.498228	0.609437	0.544311	0.189283	0.645806	0.583091	1.00000
stroke	-0.011971	0.171722	0.118664	0.186432	-0.055525	0.172785	0.211989	-0.06679
compression-ratio	-0.181258	0.247730	0.160172	0.190997	0.261160	0.155382	0.024617	0.00305
horsepower	0.072655	0.375541	0.583813	0.616779	-0.084412	0.760285	0.842691	0.56852
peak-rpm	0.230597	-0.352331	-0.280986	-0.251627	-0.264078	-0.278944	-0.219008	-0.27766
city-mpg	0.011761	-0.499126	-0.689660	-0.647099	-0.102367	-0.772171	-0.710624	-0.59195
highway-mpg	0.079514	-0.566355	-0.719324	-0.692220	-0.151188	-0.812710	-0.732138	-0.60004
price	-0.084118	0.585793	0.695331	0.754273	0.138291	0.835729	0.888942	0.54687

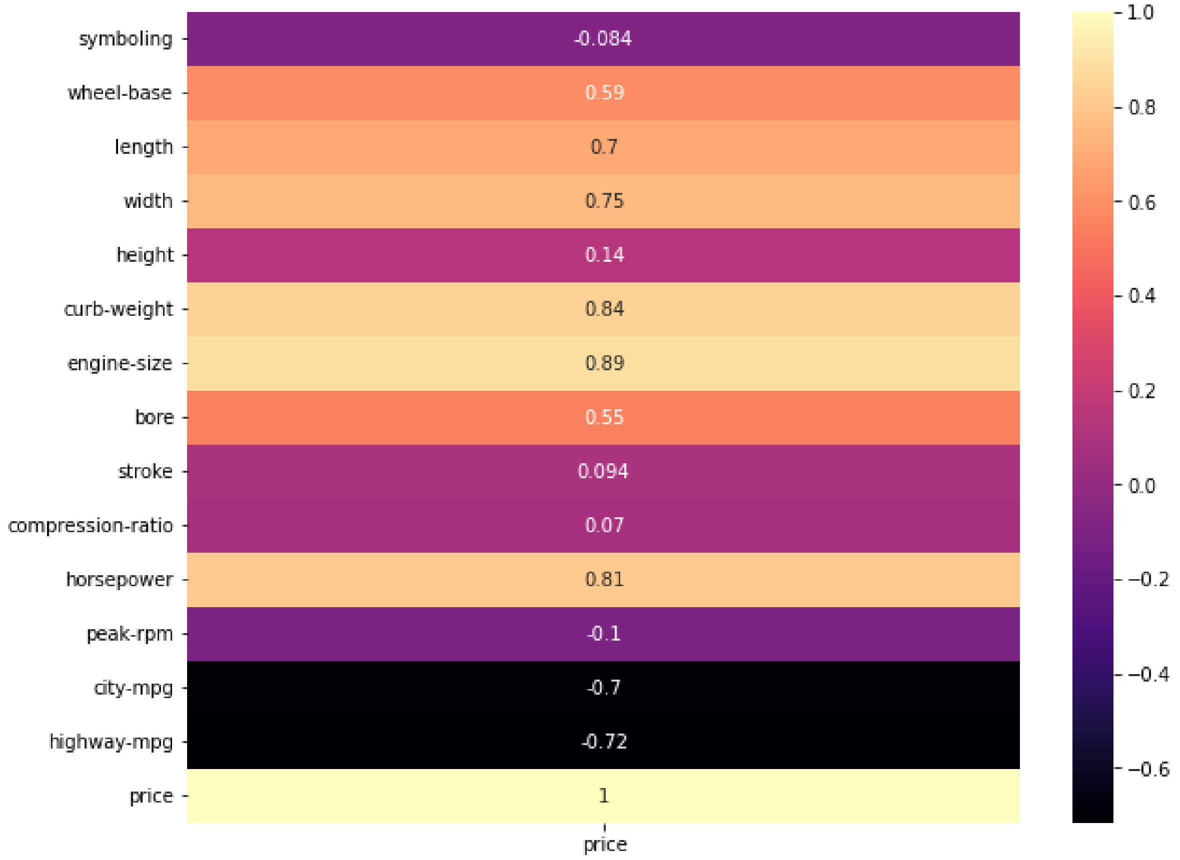
In [27]:

```python
plt.figure(figsize=(15,12))
# annot = True, anot- display values on plot
sns.heatmap(auto_price.corr(), annot=True, cmap='magma')
plt.show()
```



In [30]:

```
plt.figure(figsize=(10,8))
# annot = True, anot- display values on plot
sns.heatmap(auto_price.corr()[["price"]], annot=True, cmap='magma')
plt.show()
# This plot is used for making feature selection, we have price as our target colur
# we have to look from x asix and compare each of the feature
```


symboling
wheel-base
length
width
height
curb-weight
engine-size
bore
stroke
compression-ratio
horsepower
peak-rpm
city-mpg
highway-mpg
price
-0.084
0.59
0.7
0.75
0.14
0.84
0.89
0.55
0.094
0.07
0.81
-0.1
-0.7
-0.72
1
price
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

In [ ]: