In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns

data = pd.read_csv('C:/Users/Lenovo/Downloads/Automobile.csv')
```

Out[2]:

	symboling	normalized_losses	make	fuel_type	aspiration	number_of_doors	body_style	drive_wheels	engine_location	wheel_base	...	engine_size	fuel_system	bore	stroke	compression_ratio	horsepower	peak_rpm	c
0	3	164	audi sedan	gas	std	four	sedan	fwd	front	99.8	...	130	mpfi	3.47	2.68	9.0	111	5000	
1	3	164	audi wagon	gas	std	two	convertible	rwd	front	99.4	...	130	mpfi	3.47	2.68	9.0	111	5000	
2	1	158	audi sedan	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	
3	2	164	audi sedan	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500	
4	2	164	audi sedan	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500	

5 rows × 26 columns

In [3]:
```
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201 entries, 0 to 200
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   symboling          201 non-null    int64
 1   normalized_losses  201 non-null    int64
 2   make               201 non-null    object
 3   fuel_type          201 non-null    object
 4   aspiration         201 non-null    object
 5   number_of_doors    201 non-null    object
 6   body_style         201 non-null    object
 7   drive_wheels       201 non-null    object
 8   engine_location    201 non-null    object
 9   wheel_base         201 non-null    float64
 10  length             201 non-null    float64
 11  width              201 non-null    float64
 12  height             201 non-null    int64
 13  curb_weight        201 non-null    int64
 14  engine_type        201 non-null    object
 15  number_of_cylinders 201 non-null   object
 16  engine_size        201 non-null    int64
 17  fuel_system        201 non-null    object
 18  bore               201 non-null    float64
 19  stroke             197 non-null    float64
 20  compression_ratio  201 non-null    float64
 21  horsepower         201 non-null    int64
 22  peak_rpm           201 non-null    int64
 23  city_mpg           201 non-null    int64
 24  highway_mpg        201 non-null    int64
 25  price              201 non-null    int64
dtype: float64(7), int64(12), object(7)
memory usage: 41.8+ KB
None
```

In [4]:
```
print(data.describe)
```

```
<bound method DataFrame.describe of       symboling  normalized_losses  wheel_base    length     width  \
count  201.000000         201.000000  201.000000  201.000000  201.000000
mean     0.840796         122.160676   98.797015   174.200995   65.889055
std      1.254802          35.442168    6.066366    12.322175    2.145204
min     -2.000000          65.000000   86.600000   141.100000   60.300000
25%      0.000000         101.000000   94.500000   166.800000   64.100000
50%      1.000000         122.000000   97.000000   173.200000   65.500000
75%      2.000000         150.000000  102.400000   183.500000   66.600000
max      3.000000         256.000000  120.900000   208.100000   72.000000

       height  curb_weight  engine_size        bore      stroke  \
count  201.000000   201.000000   201.000000  201.000000  197.000000
mean    53.766667  2555.666667   126.875622    3.329751    3.255423
std      2.447822   520.680204    41.546834    0.268335    0.316726
min     47.800000  1488.000000    61.000000    2.540000    2.070000
25%     52.000000  2145.000000    98.000000    3.150000    3.110000
50%     54.100000  2414.000000   120.000000    3.310000    3.290000
75%     55.500000  2935.000000   141.000000    3.580000    3.410000
max     59.800000  4066.000000   326.000000    3.940000    4.170000

       compression_ratio  horsepower  peak_rpm  city_mpg  highway_mpg  \
count         201.000000  201.000000  201.000000  201.000000  201.000000
mean           10.164279  103.405975 5117.865672   25.179104   30.686567
std             4.004965   37.365672  478.113805    6.423220    6.815150
min             7.000000   48.000000 4150.000000   13.000000   16.000000
25%             8.600000   70.000000 4800.000000   19.000000   25.000000
50%             9.000000   95.000000 5200.000000   24.000000   30.000000
75%             9.400000  116.000000 5500.000000   30.000000   34.000000
max            23.000000  262.000000 6600.000000   49.000000   54.000000

       price
count  201.000
mean 13207.129
std   7947.066
min   5118.000
25%   7775.000
50%  10295.000
75%  16500.000
max  45400.000
```

PLOTTING UNIVARIATE DISTRIBUTIONS by using distplot() function the most easy way to take a quick look at a univariate distribution in seaborn

In [5]:
```
sns.distplot(data['normalized_losses'])
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt
your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[5]:
```
<AxesSubplot:xlabel='normalized_losses', ylabel='Density'>
```

Histogram

by using hist we remove the density curve and draw a rug plot in which vertical lick at each observation

In [6]:
```
sns.distplot(data['city_mpg'],bin = False,rug=True)
```

Out[6]:
```
<AxesSubplot:xlabel='city_mpg', ylabel='Density'>
```

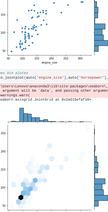
In [7]: *depicting bivariate distribution*

rewrite plots

```
sns.jointplot(data['city_mpg'],data['horsepower'])
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid position
al arguments will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[8]:
```
<seaborn.axisgrid.JointGrid at 0x2e1d1e5b2ca0>
```

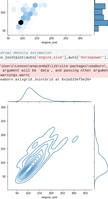


In [9]: *Make box plots*

```
sns.jointplot(data['engine_size'],data['horsepower'],kind='hex')
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid position
al arguments will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[10]:
```
<seaborn.axisgrid.JointGrid at 0x2e1d1e7af20>
```

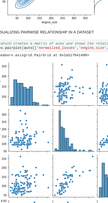


In [11]: *kernal density estimation*

```
sns.jointplot(data['engine_size'],data['horsepower'],kind='kde')
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid position
al arguments will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[12]:
```
<seaborn.axisgrid.JointGrid at 0x2e1d1e7ac20>
```



VISUALIZING PAIRWISE RELATIONSHIP IN A DATASET

In [13]: *it which creates a matrix of axes and shows the relationship for each pair of columns in a data*

```
sns.pairplot(data[['normalized_losses','engine_size','horsepower','bore']])
```

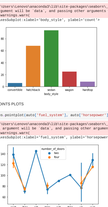
Out[14]:
```
<seaborn.axisgrid.PairGrid at 0x2e1d1f4a4d0>
```

In [15]: *depicting with categorical data*

```
sns.boxplot(data['fuel_type'],data['horsepower'])
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid position
al arguments will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[16]:
```
<AxesSubplot:xlabel='fuel_type', ylabel='horsepower'>
```



In [17] *If a strip plot, the scatterplot points will usually overlap. This makes it difficult to see the full distribution of data. One easy solution is to adjust the positions (only alo*

```
sns.stripplot(data['fuel_type'],data['horsepower'],jitter=True)
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid position
al arguments will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[18]:
```
<AxesSubplot:xlabel='fuel_type', ylabel='horsepower'>
```

In [19] *we different approach would be to use the function swarmplot(), which positions each scatterplot point on the categorical axis with an algorithm that avoids overlapping points:*

```
sns.swarmplot(data['fuel_type'],data['horsepower'])
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid position
al arguments will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\Lenovo\anaconda3\Lib\site-packages\seaborn\categorical.py:1296: UserWarning: 8.1% of the points cannot be placed; you may want to decrease the size of the markers or use s
tripplot.
  warnings.warn(msg, UserWarning)
```

Out[20]:
```
<AxesSubplot:xlabel='fuel_type', ylabel='horsepower'>
```

In [21] *Boxplot:*

```
sns.boxplot(data['number_of_doors'], data['horsepower'],hue=data['fuel_type'])
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid position
al arguments will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[22]:
```
<AxesSubplot:xlabel='number_of_doors', ylabel='horsepower'>
```


In [23] *Boxplot:*

```
sns.boxplot(data['body_style'], data['horsepower'], hue=data['engine_location'])
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid position
al arguments will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[24]:
```
<AxesSubplot:xlabel='body_style', ylabel='horsepower'>
```


In [25] *Boxplot:*

```
sns.countplot(data['body_style'])
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional
argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(msg, FutureWarning)
```

Out[26]:
```
<AxesSubplot:xlabel='count', ylabel='body_style'>
```


POINTS PLOTS

In [27]:
```
sns.pairplot(data['fuel_system'], data['horsepower'], hue=data['number_of_doors'])
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid position
al arguments will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(msg,
```

Out[28]:
```
<seaborn.axisgrid.PairGrid at 0x2e1d1f7f330>
```


DRAWING MULTI-PANNEL CATEGORICAL PLOTS

In [29]:
```
sns.factorplot(x="fuel_type",
               y="horsepower",
               hue="number_of_doors",
               data=data,kind="box",
               legend=False,
               legend_out=True)
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\seaborn\categorical.py:3296: UserWarning: 8.2% of the points cannot be placed; you may want to decrease the size of the markers or use s
tripplot.
  warnings.warn(msg, UserWarning)
```

Out[30]:
```
<seaborn.axisgrid.FacetGrid at 0x2e1d1f5f320>
```


FUNCTION TO DRAW LINEAR REGRESSION MODELS

In [31] *lmplot() is one of the best widely used function to quickly plot the linear Relationship b/w 2 variables*

```
sns.lmplot('horsepower','peak_rpm',data=data)
```

```
AttributeError                            Traceback (most recent call last)
Input In [31], in <module>----------> 1 sns.lmplot('horsepower','peak_rpm',data=data)

AttributeError: module 'seaborn' has no attribute 'lmplot'
```

In [32]:
```
sns.lmplot('horsepower', 'peak_rpm', data=data)
```

Out[33]:
```
<seaborn.axisgrid.FacetGrid at 0x2e1d1f8ac40>
```


In [34]: