Data Analysis Carsales data

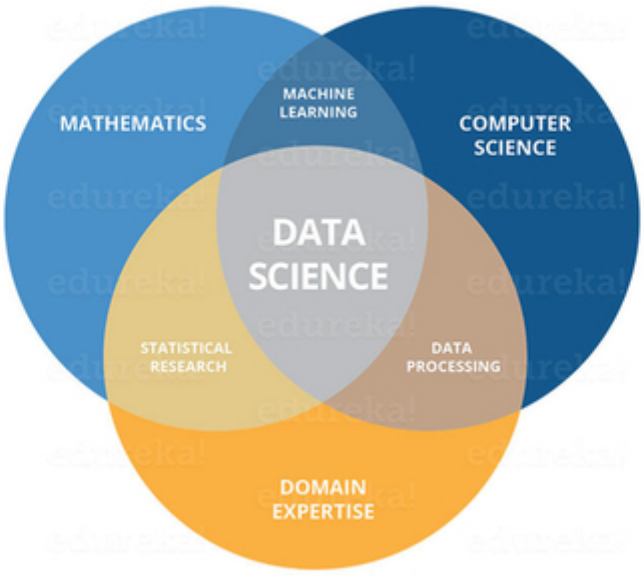
This notebooks explores the basic use of Pandas and will cover the basic commands of Exploratory Data Analysis(EDA) which includes cleaning, ploting, combining, reshaping, slicing, removing null datas, and transforming data for analysis purpose.

- The data which is used here is raw data of car sales by differnet manufacturer company where I have perform Exploratory Data Analysis.

Introduction to pandas

1. What is data science ?

Data science is the process of analysing data and extracting the useful informations.To extract the useful infomations first you need to analyse the dataset by performing some operations.
In another way you can say that it is the science of data,fiding hidden relationships and using these to predict the future.



Data science combining various fields like data preprocessing ,domain expertise, Machine Learning, Mathematics, Statistics.Here we only talk about data preprocessing in efficient way by pandas.

2. What is pandas ?

Pandas is used as data cleaning tool in the field of data science.You can do whatever operation you want in the dataset with this tool.Now question arises, can we clean or change the value in the dataset manually ? Answer is yes we can if size of the dataset is small.What if we have a large dataset then we can not do it manually it will take a lot of time.Pandas makes data science very easy and effective.

To use pandas you need to first import the pandas module in your program and other main modules

To use pandas you need to first import the pandas module in your program and other main modules numpy , matplotlib, seaborn

In [1]:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
import pandas as pd
import numpy as np
```

In [3]:

```
car=pd.read_csv('datasets/Car_sales.csv')
# returning dataframe object
car.head()
#printing dataframe df
```

Out[3]:

	Manufacturer	Model	Sales in thousands	x_4_year	Vehicle type	Price in thousands	Engine size	Horsepower	Wh
0	Acura	Integra	16.919	16.360	Passenger	21.50	1.8	140.0	
1	Acura	TL	39.384	19.875	Passenger	28.40	3.2	225.0	
2	Acura	CL	14.114	18.225	Passenger	NaN	3.2	225.0	
3	Acura	RL	8.588	29.725	Passenger	42.00	3.5	210.0	
4	Audi	A4	20.397	22.255	Passenger	23.99	1.8	150.0	

In [4]:

```
print(len(car))
# there are 96453 rows
car.head() # first five rows
```

157

Out[4]:

	Manufacturer	Model	Sales in thousands	x_4_year	Vehicle type	Price in thousands	Engine size	Horsepower	Wh
0	Acura	Integra	16.919	16.360	Passenger	21.50	1.8	140.0	
1	Acura	TL	39.384	19.875	Passenger	28.40	3.2	225.0	
2	Acura	CL	14.114	18.225	Passenger	NaN	3.2	225.0	
3	Acura	RL	8.588	29.725	Passenger	42.00	3.5	210.0	
4	Audi	A4	20.397	22.255	Passenger	23.99	1.8	150.0	

In [5]:

```
print(car['Sales in thousands'].min())
print(car['Sales in thousands'].max())
print(car['Sales in thousands'].mean())
```

```
0.11
540.561
52.99807643312102
```

In [6]:

```
car['Manufacturer'][car['Sales in thousands']==car['Sales in thousands'].min()].head(5) #Fi
```

Out[6]:

```
83    Mitsubishi
Name: Manufacturer, dtype: object
```

In [7]:

```
 car['Manufacturer'][car['Sales in thousands']==car['Sales in thousands'].max()] #Find temp
```

Out[7]:

```
56    Ford
Name: Manufacturer, dtype: object
```

DataFrame Object

DataFrame is the a main object in pandas.It is used to represent data with rows and columns(tabular or excel spreadshee like data

In [8]:

```
car.shape
```

Out[8]:

```
(157, 17)
```

Statistics of your dataframe

- we can see the statistics of our dataframe by calling describe() function
- It will print count of the columns,mean,SD,min,max etc..

In [9]:

```
car.describe()
```

Out[9]:

	Sales in thousands	x_4_year	Price in thousands	Engine size	Horsepower	Wheelbase	Width	
count	157.000000	121.000000	155.000000	156.000000	156.000000	156.000000	156.000000	15
mean	52.998076	18.072975	27.390755	3.060897	185.948718	107.487179	71.150000	18
std	68.029422	11.453384	14.351653	1.044653	56.700321	7.641303	3.451872	1
min	0.110000	5.160000	9.235000	1.000000	55.000000	92.600000	62.600000	14
25%	14.114000	11.260000	18.017500	2.300000	149.500000	103.000000	68.400000	17
50%	29.450000	14.180000	22.799000	3.000000	177.500000	107.000000	70.550000	18
75%	67.956000	19.875000	31.947500	3.575000	215.000000	112.200000	73.425000	19
max	540.561000	67.550000	85.500000	8.000000	450.000000	138.700000	79.900000	22

In [10]:

```
#To get dataframe where temperature is greater than 9
d1=car[car['Fuel efficiency']>car['Fuel efficiency'].mean()]
d1
```

2	Acura	CL	14.114	18.225	Passenger	NaN	3.2	225.0	106.9
4	Audi	A4	20.397	22.255	Passenger	23.99	1.8	150.0	102.6
7	BMW	323i	19.747	NaN	Passenger	26.99	2.5	170.0	107.3
...	...	...	...	...	...	...	...	...	...
151	Volvo	S40	16.957	NaN	Passenger	23.40	1.9	160.0	100.5
152	Volvo	V40	3.545	NaN	Passenger	24.40	1.9	160.0	100.5
153	Volvo	S70	15.245	NaN	Passenger	27.50	2.4	168.0	104.9
154	Volvo	V70	17.531	NaN	Passenger	28.80	2.4	168.0	104.9
156	Volvo	S80	18.969	NaN	Passenger	36.00	2.9	201.0	109.9

In [11]:

```
car.columns
```

Out[11]:

```
Index(['Manufacturer', 'Model', 'Sales in thousands', 'x_4_year',
       'Vehicle type', 'Price in thousands', 'Engine size', 'Horsepower',
       'Wheelbase', 'Width', 'Length', 'Curb weight', 'Fuel capacity',
       'Fuel efficiency', 'Latest Launch', 'Unnamed: 15', 'XX'],
      dtype='object')
```

Cleaning data

In [12]:

```
car.isnull().sum()
```

Out[12]:

```
Manufacturer           0
Model                  0
Sales in thousands     0
x_4_year              36
Vehicle type           0
Price in thousands     2
Engine size            1
Horsepower             1
Wheelbase              1
Width                  1
Length                 1
Curb weight            2
Fuel capacity          1
Fuel efficiency        3
Latest Launch          0
Unnamed: 15          157
XX                    36
dtype: int64
```

In [13]:

```
#Remove unnecessary columns from the data, depending on your own understandings
car.drop(["Unnamed: 15","XX"],axis=1,inplace=True)
```

In [14]:

```
car.isnull().sum()
```

Out[14]:

```
Manufacturer          0
Model                 0
Sales in thousands    0
x_4_year             36
Vehicle type          0
Price in thousands    2
Engine size           1
Horsepower            1
Wheelbase             1
Width                 1
Length                1
Curb weight           2
Fuel capacity         1
Fuel efficiency       3
Latest Launch         0
dtype: int64
```

Treat all the missing values with mean/median or most occuring observation

In [15]:

```
car["x_4_year"].fillna(value=car["x_4_year"].mean(),inplace=True)
car["Price in thousands"].fillna(value=car["Price in thousands"].mean(),inplace=True)
car["Curb weight"].fillna(value=car["Curb weight"].mean(),inplace=True)
car["Engine size"].fillna(value=car["Engine size"].mean(),inplace=True)
car["Horsepower"].fillna(value=car["Horsepower"].mean(),inplace=True)
car["Wheelbase"].fillna(value=car["Wheelbase"].mean(),inplace=True)
car["Width"].fillna(value=car["Width"].mean(),inplace=True)
car["Length"].fillna(value=car["Length"].mean(),inplace=True)
car["Fuel efficiency"].fillna(value=car["Fuel efficiency"].mean(),inplace=True)
car["Fuel capacity"].fillna(value=car["Fuel capacity"].mean(),inplace=True)
car
```

2	Acura	CL	14.114	18.225000	Passenger	27.390755	3.2	225.0	106.9
3	Acura	RL	8.588	29.725000	Passenger	42.000000	3.5	210.0	114.6
4	Audi	A4	20.397	22.255000	Passenger	23.990000	1.8	150.0	102.6
...	...	...	...	...	...	...	...	...	...
152	Volvo	V40	3.545	18.072975	Passenger	24.400000	1.9	160.0	100.5
153	Volvo	S70	15.245	18.072975	Passenger	27.500000	2.4	168.0	104.9
154	Volvo	V70	17.531	18.072975	Passenger	28.800000	2.4	168.0	104.9
155	Volvo	C70	3.493	18.072975	Passenger	45.500000	2.3	236.0	104.9
156	Volvo	S80	18.969	18.072975	Passenger	36.000000	2.9	201.0	109.9

In [16]:

```
car.isnull().sum()
```

Out[16]:

```
Manufacturer          0
Model                 0
Sales in thousands    0
x_4_year              0
Vehicle type          0
Price in thousands    0
Engine size           0
Horsepower            0
Wheelbase             0
Width                 0
Length                0
Curb weight           0
Fuel capacity         0
Fuel efficiency       0
Latest Launch         0
dtype: int64
```

In [17]:

```
car["Manufacturer"]=car["Manufacturer"].str.strip()
car["Manufacturer"].values
```

```
'Dodge', 'Dodge', 'Dodge', 'Dodge', 'Dodge', 'Dodge', 'Dodge',
'Dodge', 'Dodge', 'Dodge', 'Dodge', 'Ford', 'Ford', 'Ford', 'Ford',
'Ford', 'Ford', 'Ford', 'Ford', 'Ford', 'Ford', 'Ford', 'Honda',
'Honda', 'Honda', 'Honda', 'Honda', 'Hyundai', 'Hyundai',
'Hyundai', 'Infiniti', 'Jaguar', 'Jeep', 'Jeep', 'Jeep', 'Lexus',
'Lexus', 'Lexus', 'Lexus', 'Lexus', 'Lexus', 'Lincoln', 'Lincoln',
'Lincoln', 'Mitsubishi', 'Mitsubishi', 'Mitsubishi', 'Mitsubishi',
'Mitsubishi', 'Mitsubishi', 'Mitsubishi', 'Mercury', 'Mercury',
'Mercury', 'Mercury', 'Mercury', 'Mercury', 'Mercedes-Benz',
'Mercedes-Benz', 'Mercedes-Benz', 'Mercedes-Benz', 'Mercedes-Benz',
'Mercedes-Benz', 'Mercedes-Benz', 'Mercedes-Benz', 'Mercedes-Benz',
'Nissan', 'Nissan', 'Nissan', 'Nissan', 'Nissan', 'Nissan',
'Nissan', 'Oldsmobile', 'Oldsmobile', 'Oldsmobile', 'Oldsmobile',
'Oldsmobile', 'Oldsmobile', 'Plymouth', 'Plymouth', 'Plymouth',
'Plymouth', 'Pontiac', 'Pontiac', 'Pontiac', 'Pontiac', 'Pontiac',
'Pontiac', 'Porsche', 'Porsche', 'Porsche', 'Saab', 'Saab',
'Saturn', 'Saturn', 'Saturn', 'Saturn', 'Saturn', 'Subaru',
'Subaru', 'Toyota', 'Toyota', 'Toyota', 'Toyota', 'Toyota',
'Toyota', 'Toyota', 'Toyota', 'Toyota', 'Volkswagen', 'Volkswagen',
```

Fetch the name of the model and company's name which have lowest sales.

In [18]:

```
car.loc[:,["Manufacturer","Model","Sales in thousands"]].sort_values("Sales in thousands",a
```

Out[18]:

	Manufacturer	Model	Sales in thousands
83	Mitsubishi	3000GT	0.110
39	Dodge	Viper	0.916

Fetch the name of the model and company's name which have lowest sales with respect to Vehicle Type Passenger

In [19]:

```
car.groupby("Vehicle type").min()
```

Out[19]:

Vehicle type	Manufacturer	Model	Sales in thousands	x_4_year	Price in thousands	Engine size	Horsepower	Whee
Car	Cadillac	4Runner	9.126	7.85	11.528	2.0	119.0	
Passenger	Acura	3-Sep	0.110	5.16	9.235	1.0	55.0	

Fetch the name of the model and company's name which have Highest sales with respect to Engine Size > 6

In [20]:

```
x1=car[car["Engine size"]>6].sort_values("Sales in thousands",ascending=False)
x1.loc[:,["Manufacturer","Model","Sales in thousands","Engine size"]]
```

Out[20]:

	Manufacturer	Model	Sales in thousands	Engine size
39	Dodge	Viper	0.916	8.0

Fetch the company name who has got lowest mileage with respect to Vehicle Type as Passengers.

In [21]:

```
x2=car[car["Vehicle type"]=="Passenger"].sort_values("Fuel efficiency",ascending=True)
x2.loc[:,["Manufacturer","Model","Vehicle type","Fuel efficiency"]].head(2)
```

Out[21]:

	Manufacturer	Model	Vehicle type	Fuel efficiency
39	Dodge	Viper	Passenger	16.0
95	Mercedes-Benz	SL-Class	Passenger	20.0

Fetch the company name who has got least price and maximum number of sales figures.

In [22]:

```
x3=car.loc[:,["Manufacturer","Model","Sales in thousands","Price in thousands"]].sort_value

x3["ratio of sales/price"]=car["Sales in thousands"]/car["Price in thousands"]
x3.sort_values("ratio of sales/price",ascending=False).head(2)
```

Out[22]:

	Manufacturer	Model	Sales in thousands	Price in thousands	ratio of sales/price
56	Ford	F-Series	540.561	26.935	20.069092
55	Ford	Ranger	220.650	12.050	18.311203

Fetch the model who has got maximum number of Horsepower

In [23]:

```
x4=car.loc[:,["Model","Horsepower"]].sort_values("Horsepower",ascending=False)
x4.head(2)
```

Out[23]:

	Model	Horsepower
39	Viper	450.0
24	Corvette	345.0

Fetch the model and company name wo has got minimum mileage with highest sales figures.

In [24]:

```
x5=car.loc[:,["Manufacturer","Model","Sales in thousands","Fuel efficiency"]].sort_values("
x5.head(2)
```

Out[24]:

	Manufacturer	Model	Sales in thousands	Fuel efficiency
78	Lincoln	Navigator	22.925	15.0
144	Toyota	Land Cruiser	9.835	15.0

- Fetch the model and company name where the wheel base is maximum with Vehicle type as Car

In [25]:

```
x6=car[car["Vehicle type"]=="Car"].sort_values("Wheelbase",ascending=False)
x6.loc[:,["Manufacturer","Model","Vehicle type","Wheelbase"]].head(5)
```

Out[25]:

	Manufacturer	Model	Vehicle type	Wheelbase
40	Dodge	Ram Pickup	Car	138.7
56	Ford	F-Series	Car	138.5
43	Dodge	Dakota	Car	131.0
42	Dodge	Ram Van	Car	127.2
53	Ford	Windstar	Car	120.7

Fetch the model and company name where the wheel base is minimum with Vehicle type as Passengers

In [26]:

```
x7=car[car["Vehicle type"]=="Passenger"].sort_values("Wheelbase",ascending=True)
x7.loc[:,["Manufacturer","Model","Vehicle type","Wheelbase"]].head(5)
```

Out[26]:

	Manufacturer	Model	Vehicle type	Wheelbase
126	Porsche	Carrera Cabriolet	Passenger	92.6
125	Porsche	Carrera Coupe	Passenger	92.6
26	Chevrolet	Metro	Passenger	93.1
96	Mercedes-Benz	SLK	Passenger	94.5
97	Mercedes-Benz	SLK230	Passenger	94.5

Fetch the model and company name where the curb weight is minimum with minimum mileage

In [27]:

```
x8=car.loc[:,["Manufacturer","Model","Sales in thousands","Curb weight","Fuel efficiency"]]
x8.head()
```

Out[27]:

	Manufacturer	Model	Sales in thousands	Curb weight	Fuel efficiency
26	Chevrolet	Metro	21.855	1.895	45.0
62	Hyundai	Accent	41.184	2.240	31.0
79	Mitsubishi	Mirage	26.232	2.250	30.0
129	Saturn	SL	80.620	2.332	33.0
57	Honda	Civic	199.685	2.339	32.0

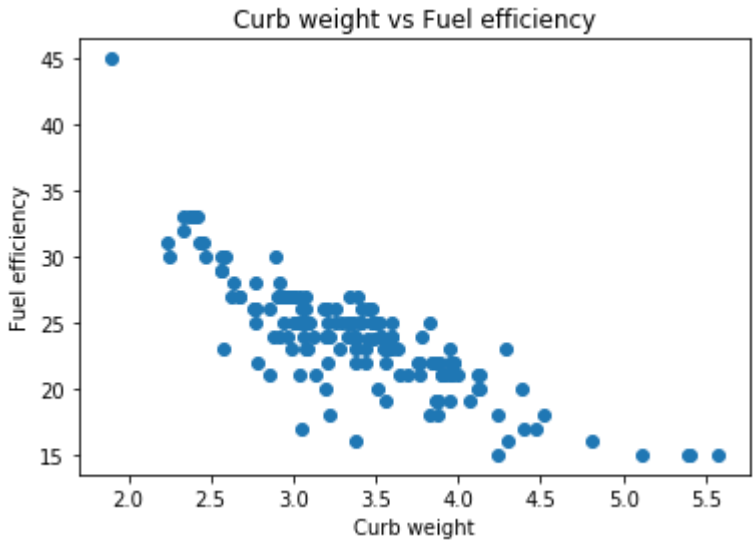
Matplotlib -scatter plot

In [28]:

```
plt.scatter(car["Curb weight"],car["Fuel efficiency"])
plt.xlabel("Curb weight")
plt.ylabel("Fuel efficiency")
plt.title("Curb weight vs Fuel efficiency")
```

Out[28]:

```
Text(0.5, 1.0, 'Curb weight vs Fuel efficiency')
```



Change the name of column x_4_year to 4_Year_Resale_Value

In [29]:

```
car.rename(columns={ "x_4_year":"4_Year_Resale_Value"},inplace=True)
car.head()
```

Out[29]:

	Manufacturer	Model	Sales in thousands	4_Year_Resale_Value	Vehicle type	Price in thousands	Engine size	Hors
0	Acura	Integra	16.919	16.360	Passenger	21.500000	1.8	
1	Acura	TL	39.384	19.875	Passenger	28.400000	3.2	
2	Acura	CL	14.114	18.225	Passenger	27.390755	3.2	
3	Acura	RL	8.588	29.725	Passenger	42.000000	3.5	
4	Audi	A4	20.397	22.255	Passenger	23.990000	1.8	

Fetch the model and company name whose sales and Resale Value is Highest

In [30]:

```
car.groupby("Manufacturer").max().sort_values("Sales in thousands",ascending=False).head().
```

Out[30]:

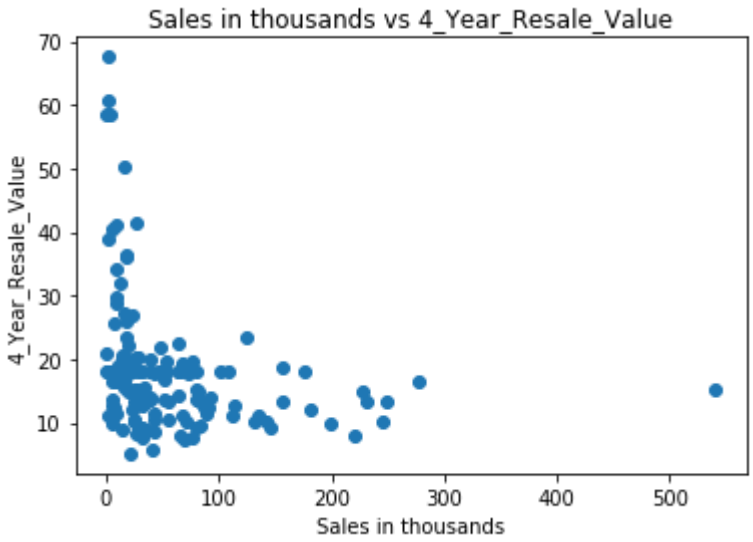
Manufacturer	Model	Sales in thousands	4_Year_Resale_Value
Ford	Windstar	540.561	23.575
Toyota	Tacoma	247.994	34.080
Honda	Passport	230.902	19.490
Dodge	Viper	227.061	58.470
Jeep	Wrangler	157.040	18.810

In [31]:

```
plt.scatter(car["Sales in thousands"],car["4_Year_Resale_Value"])
plt.xlabel("Sales in thousands")
plt.ylabel("4_Year_Resale_Value")
plt.title("Sales in thousands vs 4_Year_Resale_Value")
```

Out[31]:

```
Text(0.5, 1.0, 'Sales in thousands vs 4_Year_Resale_Value')
```



Fetch the model and company name whose sales, Mileage and Resale Value is Highest but Price is lowest

In [32]:

```
car.groupby("Manufacturer").min().sort_values("Price in thousands",ascending=True).head(5).
```

Out[32]:

Manufacturer	Model	Sales in thousands	Price in thousands	Fuel capacity	4_Year_Resale_Value
Chevrolet	Camaro	17.947	9.235	10.3	5.160
Hyundai	Accent	29.450	9.699	11.9	5.860
Saturn	LS	5.223	10.685	12.1	9.200
Toyota	4Runner	9.835	11.528	13.2	9.575
Ford	Contour	35.068	12.050	12.7	7.425

Fetch the model and company name whole resale value is lowest but sale is highest.

In [33]:

```
car.groupby("Manufacturer").min().sort_values("4_Year_Resale_Value",ascending=True).head(3)
```

Out[33]:

Manufacturer	Model	Sales in thousands	Price in thousands	Fuel capacity	4_Year_Resale_Value
Chevrolet	Camaro	17.947	9.235	10.3	5.160
Hyundai	Accent	29.450	9.699	11.9	5.860
Ford	Contour	35.068	12.050	12.7	7.425

Matplotlib is the "grandfather" library of data visualization with Python. It was created by John Hunter. He created it to try to replicate MatLab's (another programming language) plotting capabilities in Python. So if you happen to be familiar with matlab, matplotlib will feel natural to you.

It is an excellent 2D and 3D graphics library for generating scientific figures.

Some of the major Pros of Matplotlib are:

- Generally easy to get started for simple plots
- Support for custom labels and texts
- Great control of every element in a figure
- High-quality output in many formats
- Very customizable in general

Distplot

Below we can observe the following points from distplot graph:

- average sale or maximum density for sale is approx 52k
- density of sales decreses gradually after 250k and hence can be consider outliers
- most of the cars have fuel efficiency average of approx 23
- avaerage price of cars is 27k

In [34]:

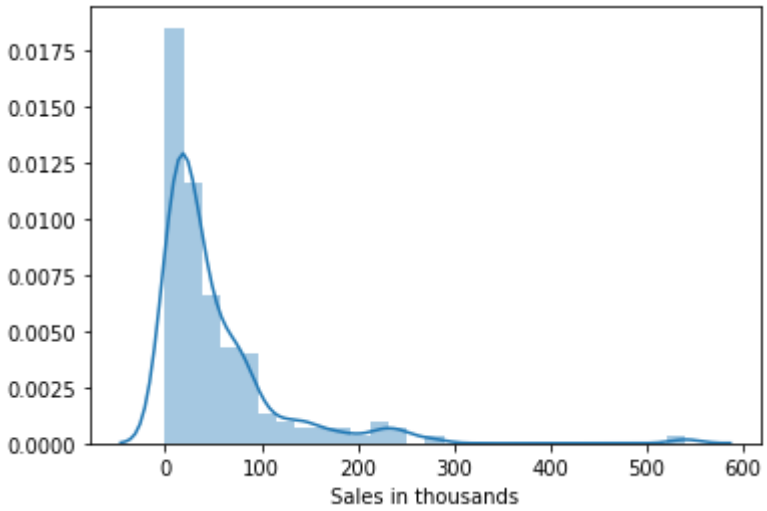
```
car['Sales in thousands'].mean()
```

Out[34]:

```
52.99807643312102
```

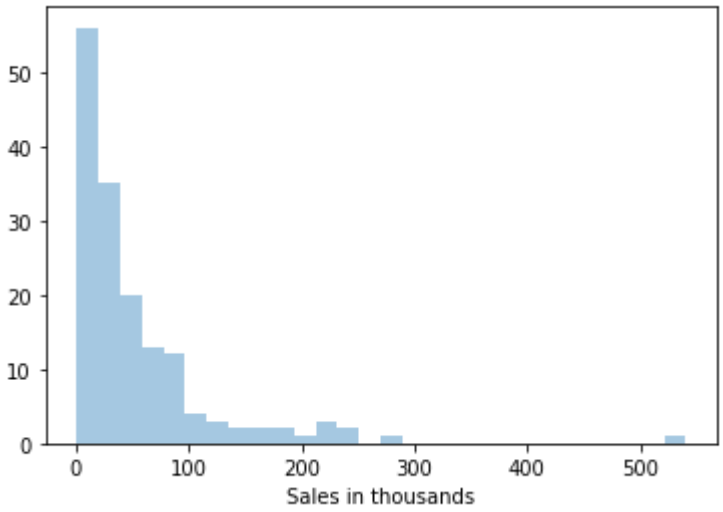
In [35]:

```
sns.distplot(car['Sales in thousands']);
```



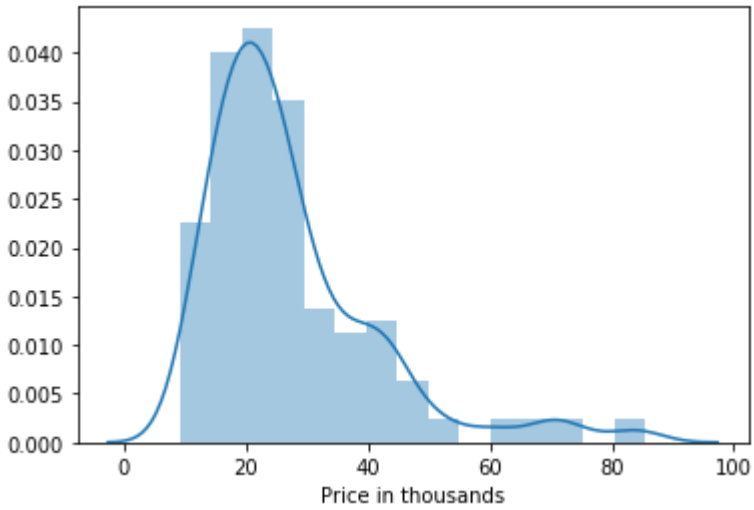
In [36]:

```
sns.distplot(car['Sales in thousands'],kde=False);
```



In [37]:

```
sns.distplot(car['Price in thousands']);
```



In [38]:

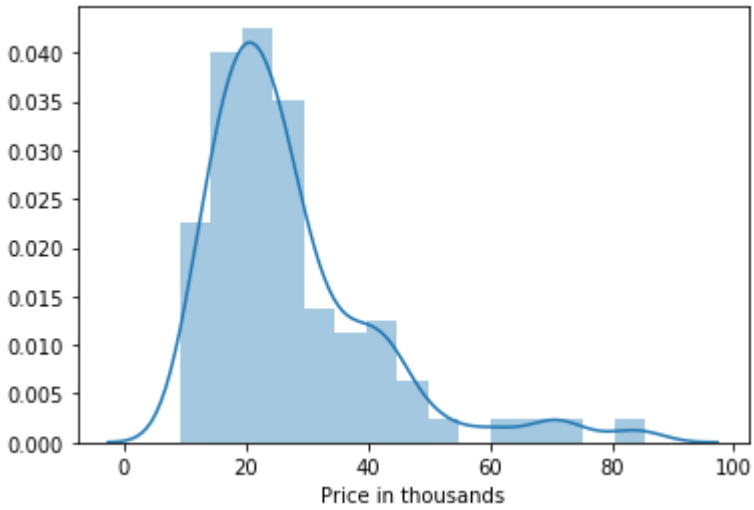
```
car['Fuel efficiency'].mean()
```

Out[38]:

```
23.844155844155846
```

In [39]:

```
sns.distplot(car['Price in thousands'],kde=True);
```



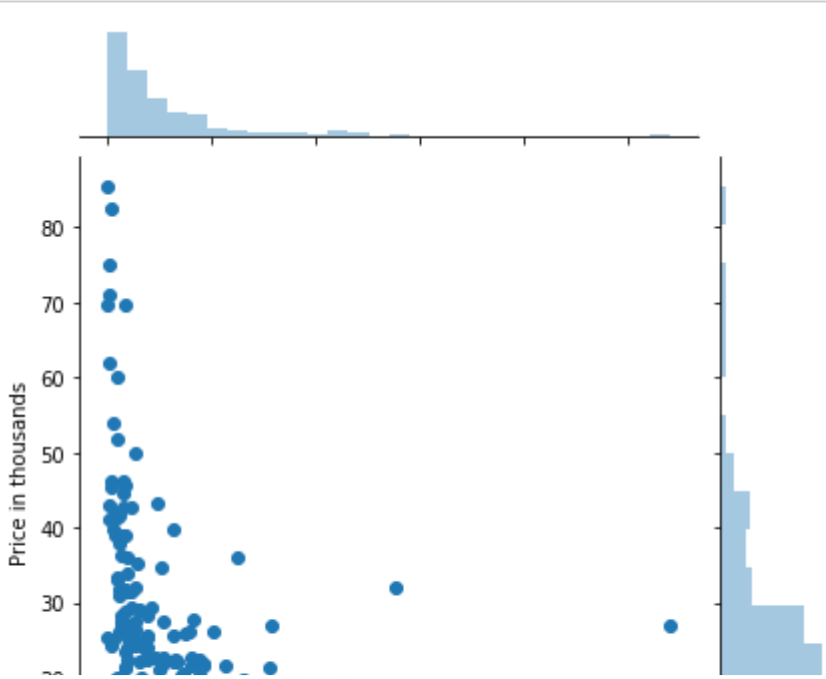
Jointplot

Below we can observe the following points from Jointplot graph:

- There is more number of sales whose price of car is below 52k
- car with good fuel efficiency and with less price has been sold more

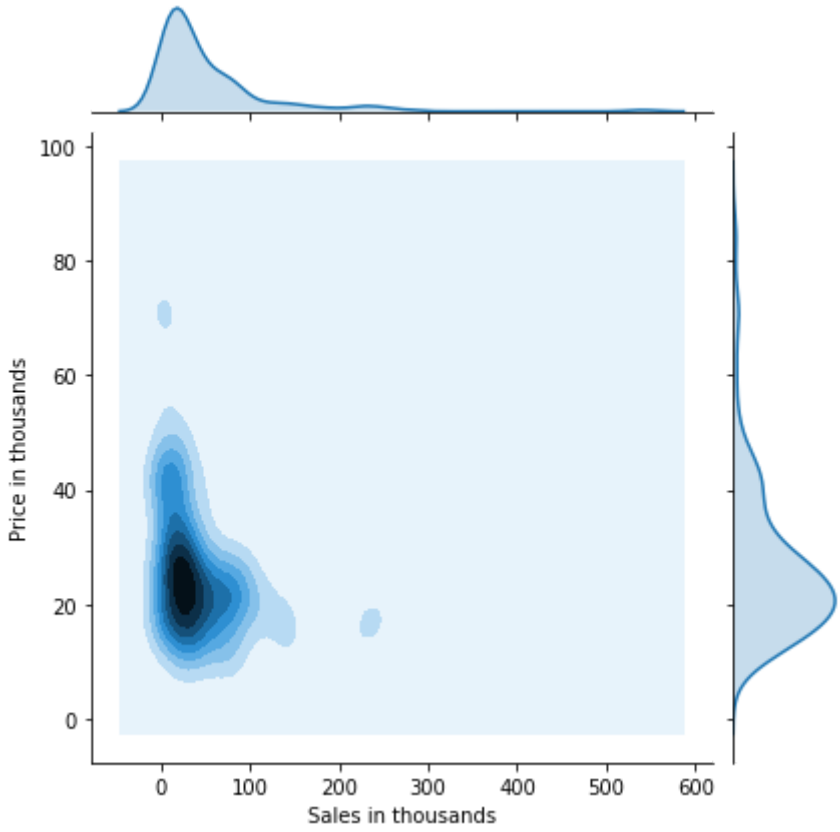
In [40]:

```
sns.jointplot(x='Sales in thousands',y='Price in thousands',data=car,kind='scatter');
```



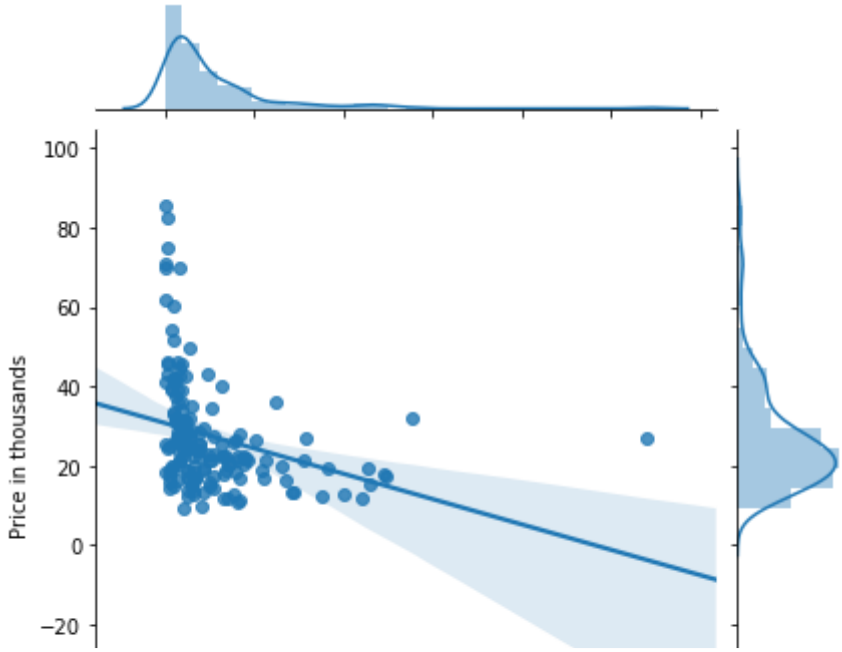
In [41]:

```
sns.jointplot(x='Sales in thousands',y='Price in thousands',data=car,kind='kde');
```



In [42]:

```
sns.jointplot(x='Sales in thousands',y='Price in thousands',data=car,kind='reg');
```



PairPlot

In [43]:

```
sns.pairplot(car);
```

Kdeplots

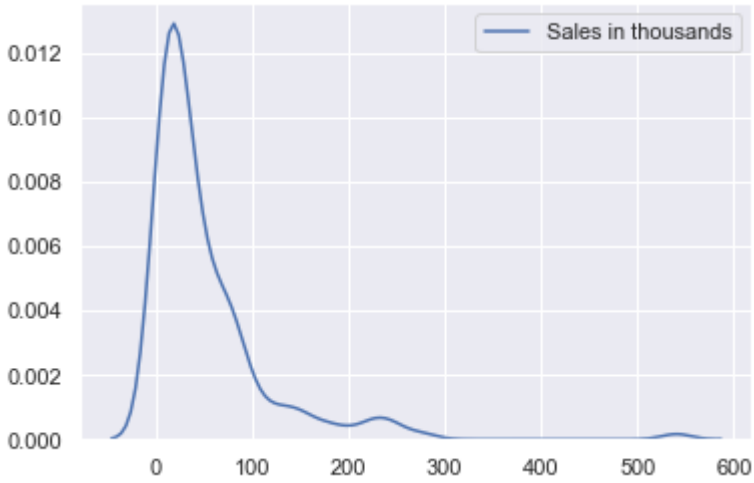
Kdeplots just only displays the normal distribution of the variables. we could also combineKdeplot with rugplot

In [44]:

```
sns.set()#background grid
sns.kdeplot(car['Sales in thousands'])
```

Out[44]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ec98147208>
```

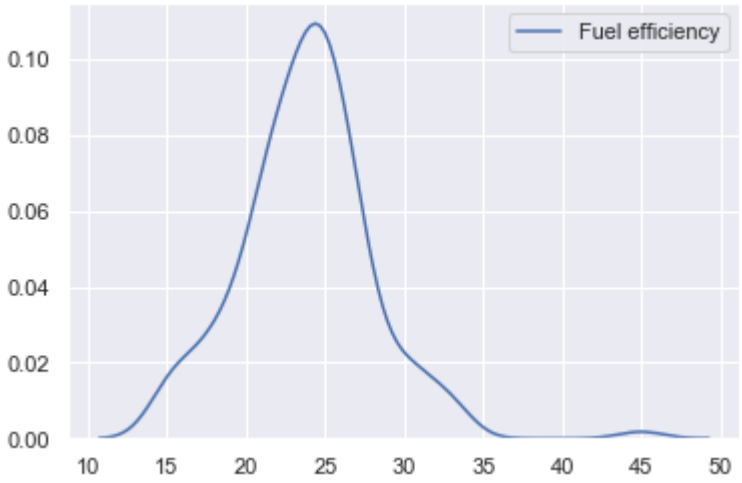


In [45]:

```
sns.set()
sns.kdeplot(car['Fuel efficiency'])
```

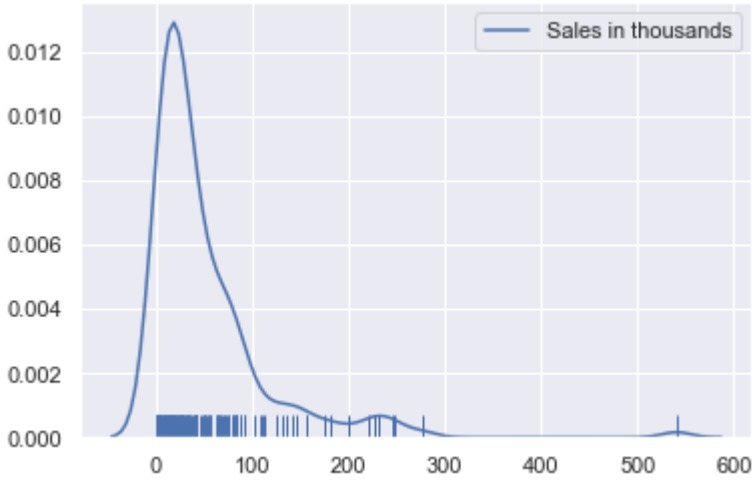
Out[45]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ec98308308>
```



In [46]:

```
sns.set()
sns.kdeplot(car['Sales in thousands']);
sns.rugplot(car['Sales in thousands']);
```



In [47]:

```
sns.set()
sns.kdeplot(car['Price in thousands']);
sns.rugplot(car['Price in thousands']);
```

A heatmap of pivot table with cars data where the pivot table will be representing at which year which company generates what sales /price figures

In [48]:

```
car["Time"]=pd.to_datetime(car["Latest Launch"])
```

In [49]:

```
car["year"]=car['Time'].dt.year
car["month"]=car['Time'].dt.month_name()
car
```

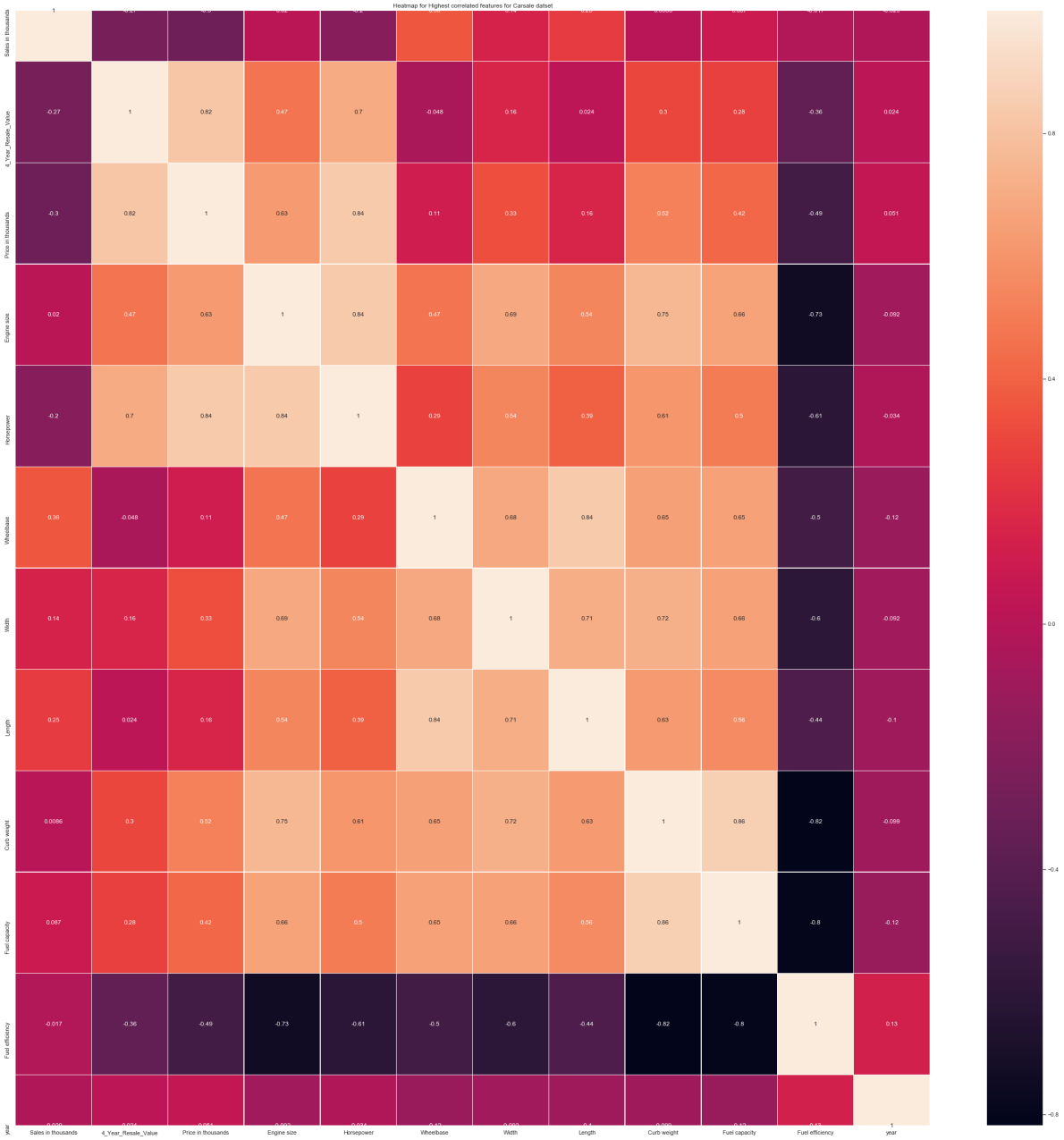
2	Acura	CL	14.114	18.225000	Passenger	27.390755	3.2	225.0
3	Acura	RL	8.588	29.725000	Passenger	42.000000	3.5	210.0
4	Audi	A4	20.397	22.255000	Passenger	23.990000	1.8	150.0
...	...	...	...	...	...	...	...	...
152	Volvo	V40	3.545	18.072975	Passenger	24.400000	1.9	160.0
153	Volvo	S70	15.245	18.072975	Passenger	27.500000	2.4	168.0
154	Volvo	V70	17.531	18.072975	Passenger	28.800000	2.4	168.0
155	Volvo	C70	3.493	18.072975	Passenger	45.500000	2.3	236.0
156	Volvo	S80	18.969	18.072975	Passenger	36.000000	2.9	201.0

In [50]:

```
plt.figure(figsize=(40,40))
sns.heatmap(car.corr(),annot=True, linewidths=.5)
plt.title("Heatmap for Highest correlated features for Carsale datset")
```

Out[50]:

```
Text(0.5, 1, 'Heatmap for Highest correlated features for Carsale datset')
```



Above graph shows the which features are most relative/correlated and dependent on each other. Hence it looks sales in thousand and wheelbase are higly correlated to each other and price may change (increase/decrease depending upon horsepower)

In [51]:

```
carsales=car.pivot_table(values='Sales in thousands',index='Manufacturer',columns='year')
carsales
```

Out[51]:

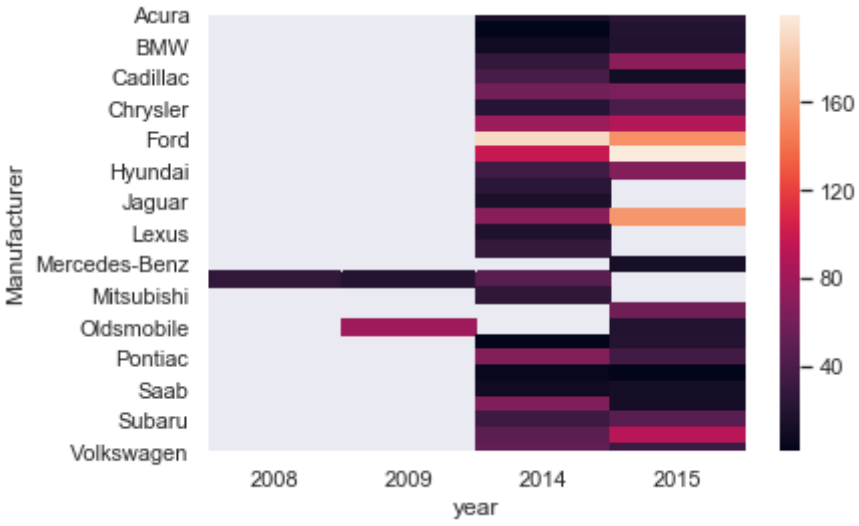
year	2008	2009	2014	2015
Manufacturer				
Acura	NaN	NaN	15.516500	23.986000
Audi	NaN	NaN	1.380000	19.588500
BMW	NaN	NaN	9.231000	18.637000
Buick	NaN	NaN	27.851000	71.389333
Cadillac	NaN	NaN	39.257000	11.221333
Chevrolet	NaN	NaN	58.309333	63.239500
Chrysler	NaN	NaN	20.790000	39.520333
Dodge	NaN	NaN	78.158500	88.239600
Ford	NaN	NaN	190.379667	154.609000

In [52]:

```
sns.heatmap(carsales)
```

Out[52]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ec9b4bf748>
```



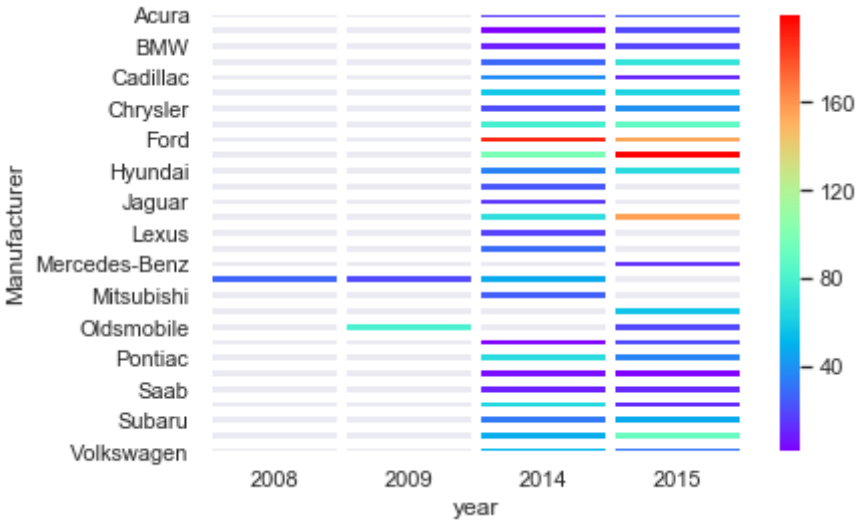
Above graph shows in which Year was the highest registrations, and hence shows max sale was done in the Year 2015 by ford. This info can be use to start working/research why sale was max in this year.

In [53]:

```
sns.heatmap(carsales,cmap='rainbow',linecolor='white',linewidths=4)
```

Out[53]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ec9947cdc8>
```



Categorical Data Plots

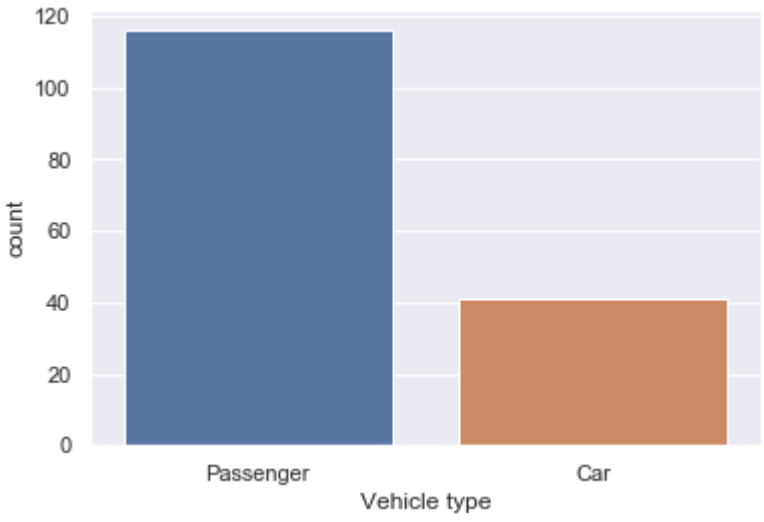
- using seaborn to plot categorical data!

In [54]:

```
sns.countplot(x='Vehicle type',data=car)
```

Out[54]:

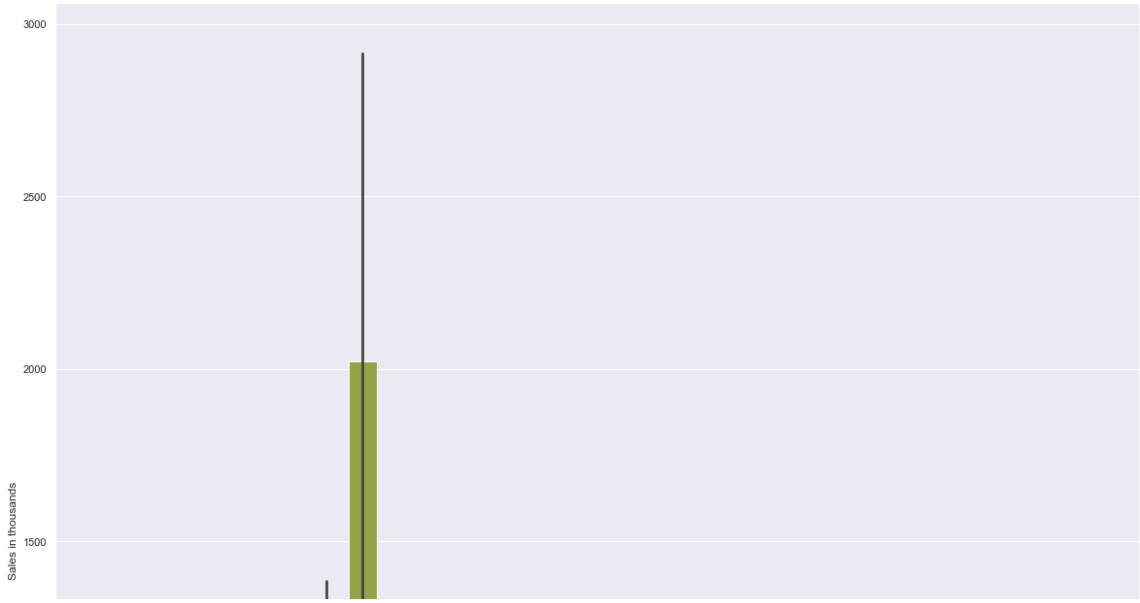
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ec99520108>
```



In [55]:

```
plt.figure(figsize=(20,20))
sns.barplot(x="Manufacturer",y='Sales in thousands',data=car,estimator=np.sum);
plt.xticks(rotation=90)
```

```
 <a list of 30 Text xticklabel objects>)
```



Below we can observe the following points from countplot graph:

- It shows the number of cars sold by all manufacturer

In [56]:

```
plt.figure(figsize=(20,20))
sns.countplot(x='Manufacturer',data=car)
plt.xticks(rotation=90)
```

```
 <a list of 30 Text xticklabel objects>)
```

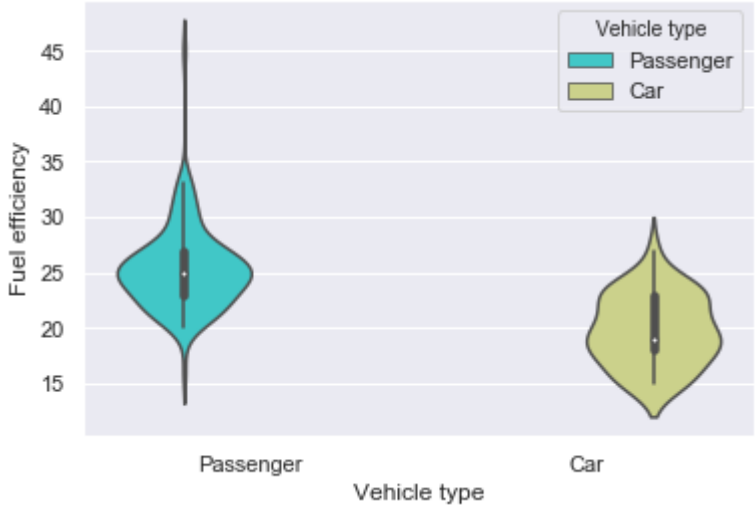
This shows ford and dodge are top most brands on sale and hence these would be preferred choices for most people

In [57]:

```
sns.violinplot(x="Vehicle type",y="Fuel efficiency",data=car,hue="Vehicle type",palette="ra
```

Out[57]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ec99760088>
```



Saving figures

Matplotlib can generate high-quality output in a number formats, including PNG, JPG, EPS, SVG, PGF and PDF.

In [58]:

```
#fig.savefig("PLOT1.jpeg")
```

OBSERVATION & CONCLUSION

- With the help of notebook I learnt how exploratory data analysis can be carried out using Pandas plotting.
- Also I have seen making use of packages like matplotlib and seaborn to develop better insights about the data.
- I have also seen how preproceesing helps in dealing with missing values and irregualities present in the data.
- I also learnt how to create new features which will in turn help us to better predict the survival.
- I also make use of pandas profiling feature to generate an html report containing all the information of the various features present in the dataset.
- I have seen the impact of columns like mileage, year and Fuel efficiency on the Price increase/decrease rate.
- The most important inference drawn from all this analysis is, I get to know what are the features on which price is highly positively and negatively coorelated with.
- This analysis will help me to choose which machine learning model we can apply to predict price of test dataset in later terms and projects.

Thank you!!!