Importing libraries that are required

In [98]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib as mp
```

Reading data

In [99]:
```
car = pd.read_csv("D:\PROJECTS\Dataset\data.csv")
```

In [100]:
```
print("No of data points and columns : ",car.shape)
```

```
No of data points and columns :  (11914, 16)
```

In [101]:
```
print("Column names : " , car.columns)
```

```
Column names :  Index(['Make', 'Model', 'Year', 'Engine Fuel Type', 'Engine H
P',
       'Engine Cylinders', 'Transmission Type', 'Driven_Wheels',
       'Number of Doors', 'Market Category', 'Vehicle Size', 'Vehicle Style',
       'highway MPG', 'city mpg', 'Popularity', 'MSRP'],
      dtype='object')
```

changing columns names

In [102]:
```
car.columns=['Make', 'Model', 'Year', 'Engine Fuel Type', 'HP',
       'Cylinders', 'Type', 'DriveMode',
       'Number of Doors', 'Market Category', 'Vehicle Size', 'Vehicle Style',
       'MPG_H', 'MPG_c', 'Popularity', 'price']
```

Observing top 5 and bottom 5

In [103]:
```
car.head(5)
```

Out[103]:

	Make	Model	Year	Engine Fuel Type	HP	Cylinders	Type	DriveMode	Number of Doors	Market Cate
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0	Fa Tuner,Luxury,I Perform
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Perform
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,I Perform
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Perform
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Lu

In [104]:
```
car.tail(5)
```

Out[104]:

	Make	Model	Year	Engine Fuel Type	HP	Cylinders	Type	DriveMode	Number of Doors
11909	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0
11910	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0
11911	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0
11912	Acura	ZDX	2013	premium unleaded (recommended)	300.0	6.0	AUTOMATIC	all wheel drive	4.0
11913	Lincoln	Zephyr	2006	regular unleaded	221.0	6.0	AUTOMATIC	front wheel drive	4.0

In [105]:
```
Makers=car['Make'].value_counts()
MP=car[['Make','price']]
print(MP)
```

```
          Make  price
0          BMW  46135
1          BMW  40650
2          BMW  36350
3          BMW  29450
4          BMW  34500
...        ...    ...
11909    Acura  46120
11910    Acura  56670
11911    Acura  50620
11912    Acura  50920
11913  Lincoln  28995

[11914 rows x 2 columns]
```

Info about the data frame(with data types of each feature)

In [106]:
```
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11914 entries, 0 to 11913
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Make              11914 non-null  object 
 1   Model             11914 non-null  object 
 2   Year              11914 non-null  int64  
 3   Engine Fuel Type  11911 non-null  object 
 4   HP                11845 non-null  float64
 5   Cylinders         11884 non-null  float64
 6   Type              11914 non-null  object 
 7   DriveMode         11914 non-null  object 
 8   Number of Doors   11908 non-null  float64
 9   Market Category   8172 non-null   object 
 10  Vehicle Size      11914 non-null  object 
 11  Vehicle Style     11914 non-null  object 
 12  MPG_H             11914 non-null  int64  
 13  MPG_c             11914 non-null  int64  
 14  Popularity        11914 non-null  int64  
 15  price             11914 non-null  int64  
dtypes: float64(3), int64(5), object(8)
memory usage: 1.5+ MB
```

Staticstics of features

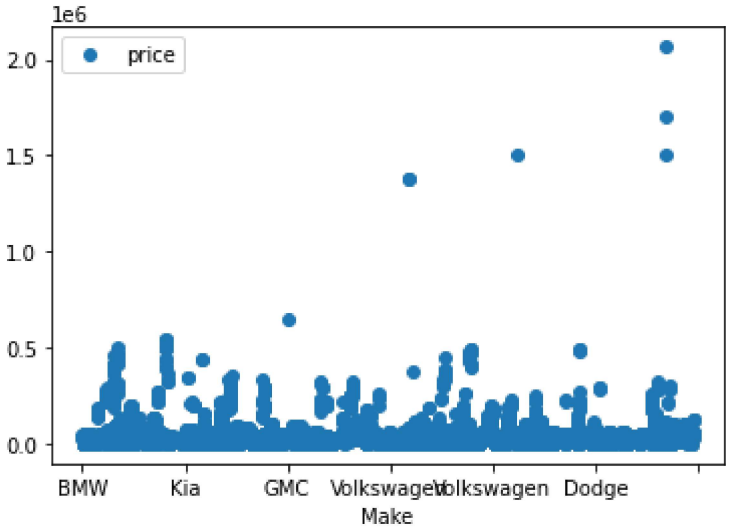
In [107]:
```
car.describe()
```

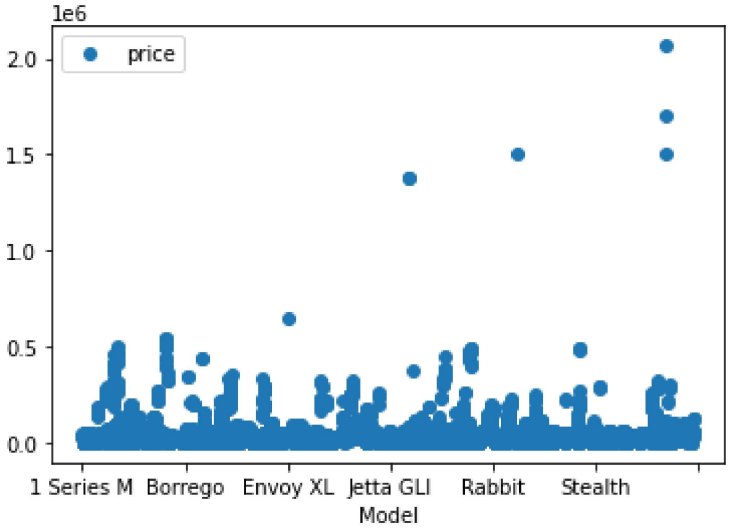
Out[107]:

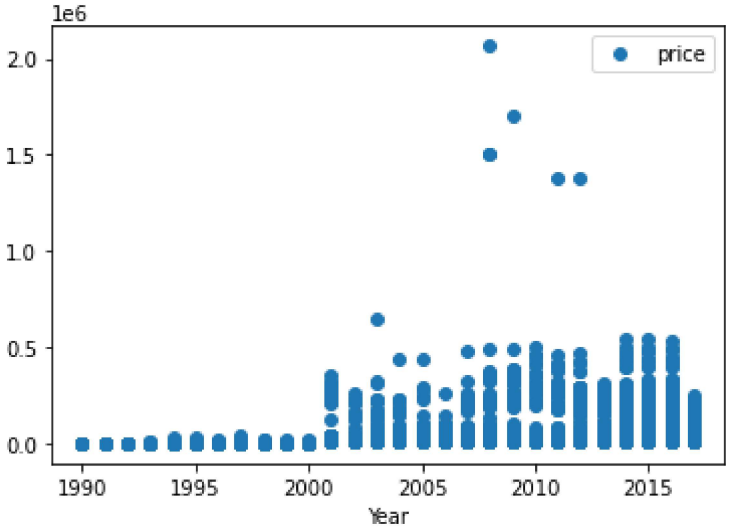
	Year	HP	Cylinders	Number of Doors	MPG_H	MPG_c	Po
count	11914.000000	11845.00000	11884.000000	11908.000000	11914.000000	11914.000000	11914
mean	2010.384338	249.38607	5.628829	3.436093	26.637485	19.733255	1554
std	7.579740	109.19187	1.780559	0.881315	8.863001	8.987798	1441
min	1990.000000	55.00000	0.000000	2.000000	12.000000	7.000000	2
25%	2007.000000	170.00000	4.000000	2.000000	22.000000	16.000000	549
50%	2015.000000	227.00000	6.000000	4.000000	26.000000	18.000000	1385
75%	2016.000000	300.00000	6.000000	4.000000	30.000000	22.000000	2009
max	2017.000000	1001.00000	16.000000	4.000000	354.000000	137.000000	5657

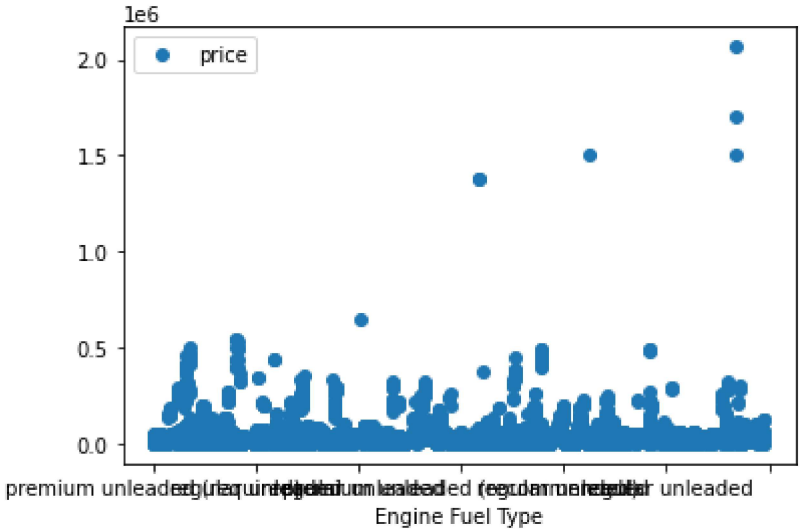
Plots to eliminates unwanted features

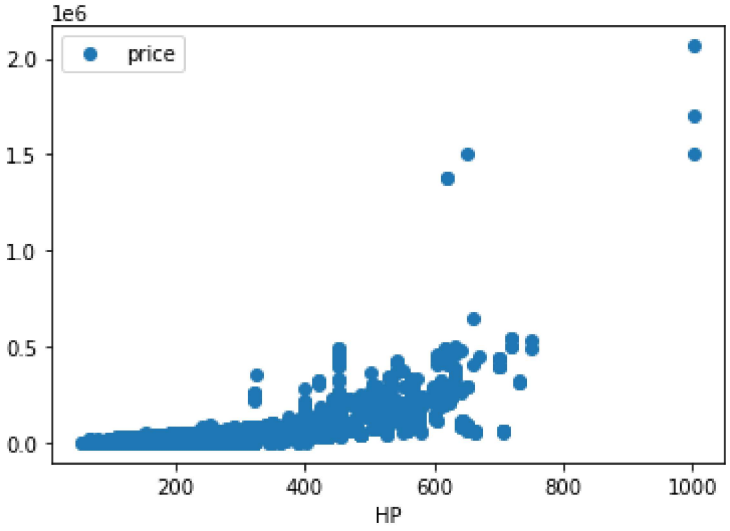
In [108]:
```
for i in car.columns:
    if(i!='price'):
        car.plot(x=i,y='price',style='o')
        #car.plot(x=i,y='price')
```

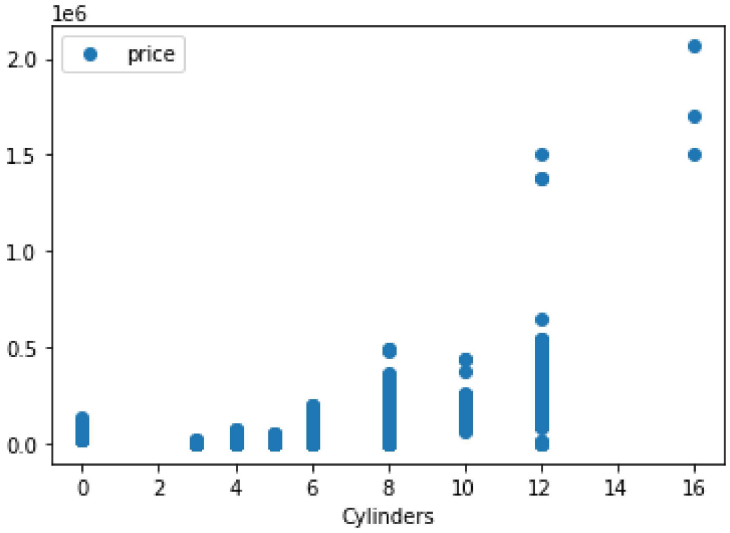

1e6
price
2.0
1.5
1.0
0.5
0.0
BMW
Kia
GMC
Volkswagen
Volkswagen
Dodge
Make

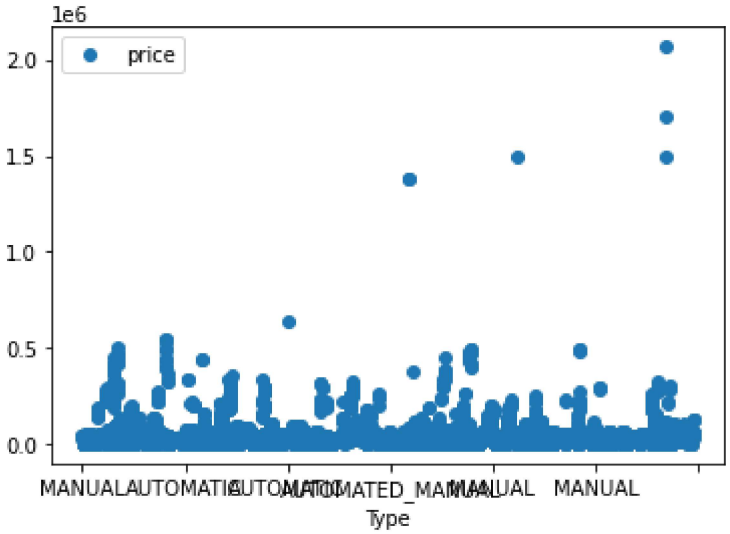

1e6
price
2.0
1.5
1.0
0.5
0.0
1 Series M
Borrego
Envoy XL
Jetta GLI
Rabbit
Stealth
Model

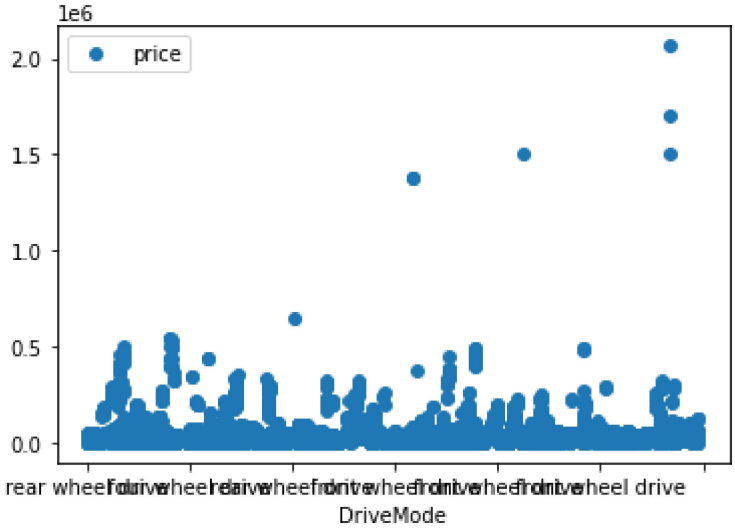

1e6
price
2.0
1.5
1.0
0.5
0.0
1990
1995
2000
2005
2010
2015
Year

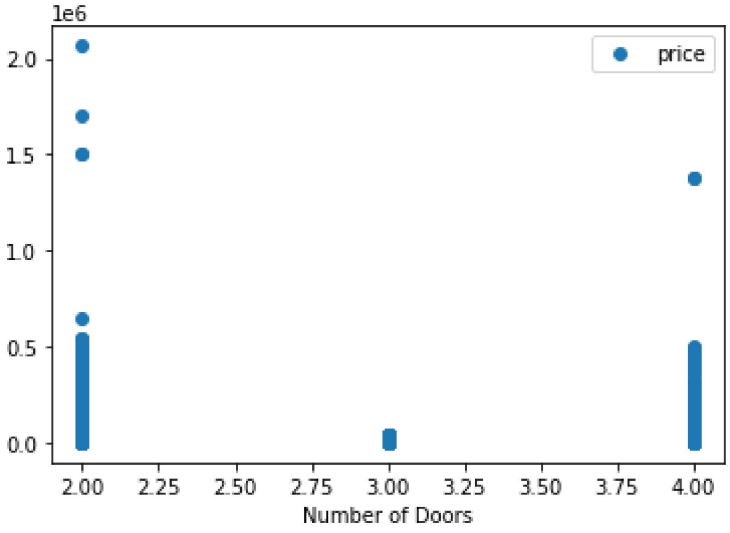

1e6
price
2.0
1.5
1.0
0.5
0.0
premium unleaded
unleaded
Engine Fuel Type

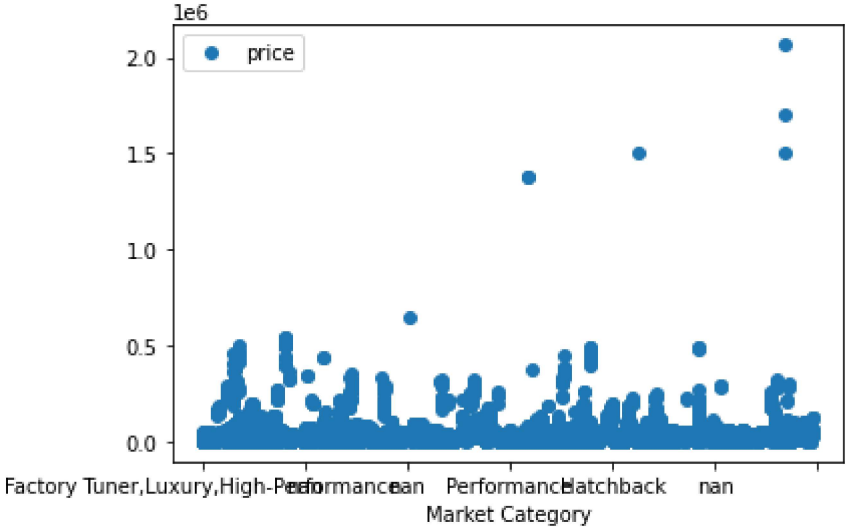

1e6
price
2.0
1.5
1.0
0.5
0.0
200
400
600
800
1000
HP

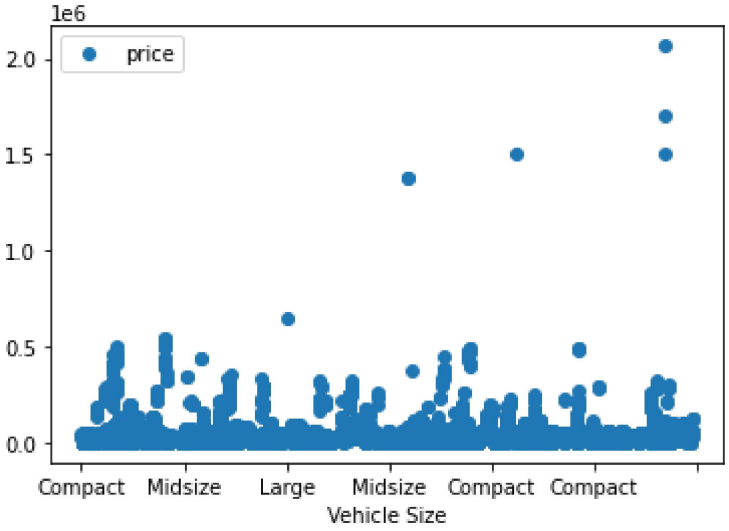

1e6
price
2.0
1.5
1.0
0.5
0.0
0
2
4
6
8
10
12
14
16
Cylinders

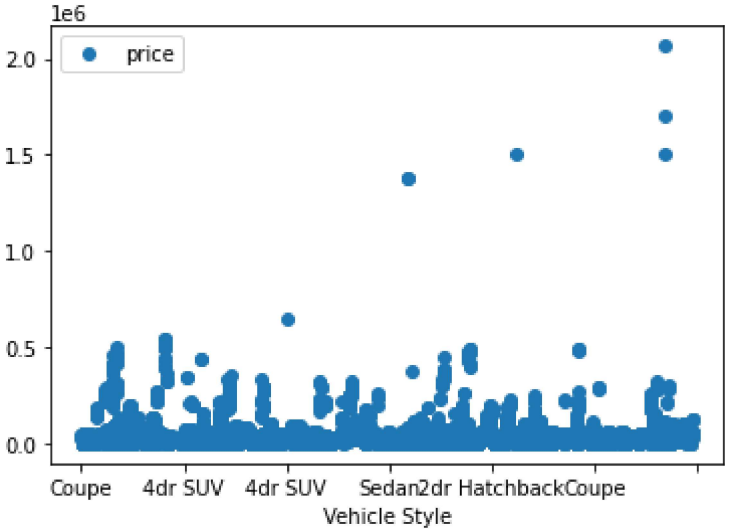

1e6
price
2.0
1.5
1.0
0.5
0.0
MANUALAUTOMATIC
MANUAL
Type

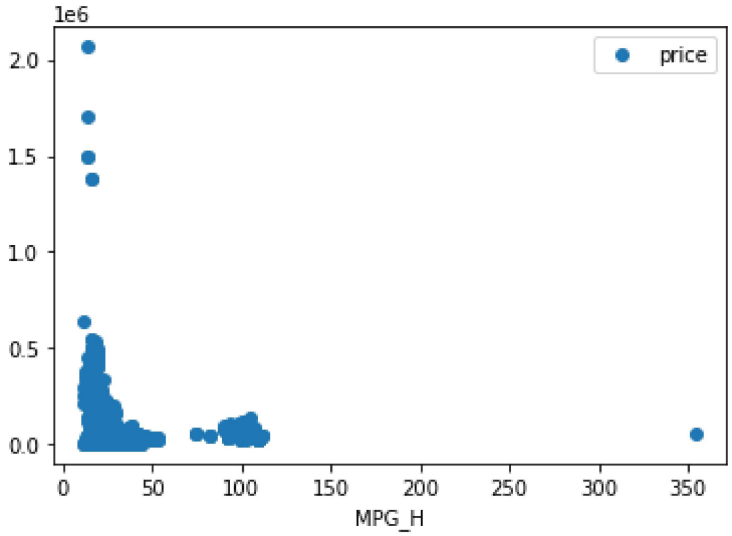

1e6
price
2.0
1.5
1.0
0.5
0.0
rear wheel drive
DriveMode

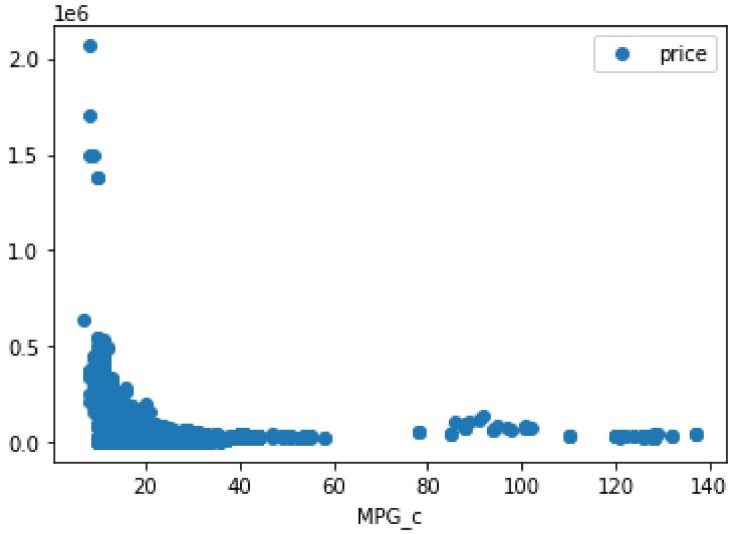

1e6
price
2.0
1.5
1.0
0.5
0.0
2.00
2.25
2.50
2.75
3.00
3.25
3.50
3.75
4.00
Number of Doors

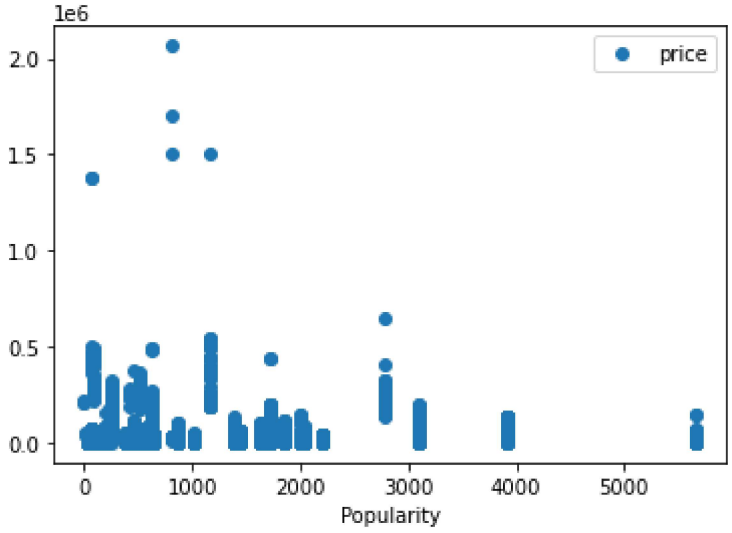

1e6
price
2.0
1.5
1.0
0.5
0.0
Factory Tuner,Luxury,High-Performance
nan
Performance
Hatchback
nan
Market Category


1e6
price
2.0
1.5
1.0
0.5
0.0
Compact
Midsize
Large
Midsize
Compact
Compact
Vehicle Size

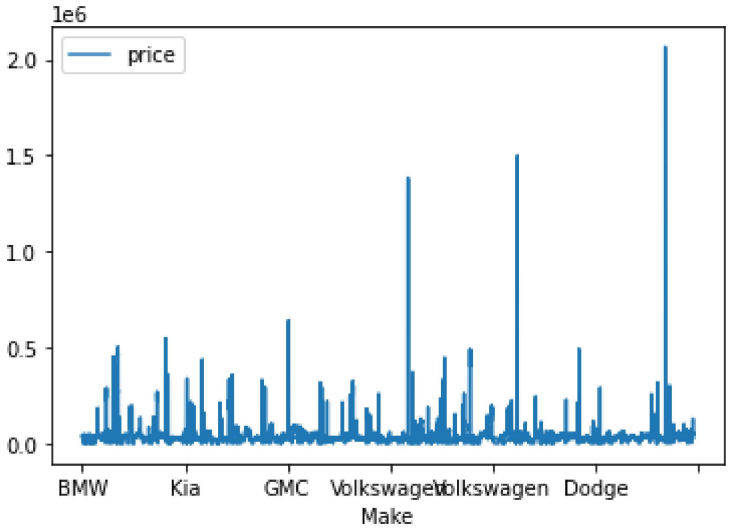

1e6
price
2.0
1.5
1.0
0.5
0.0
Coupe
4dr SUV
4dr SUV
Sedan
2dr Hatchback
Coupe
Vehicle Style

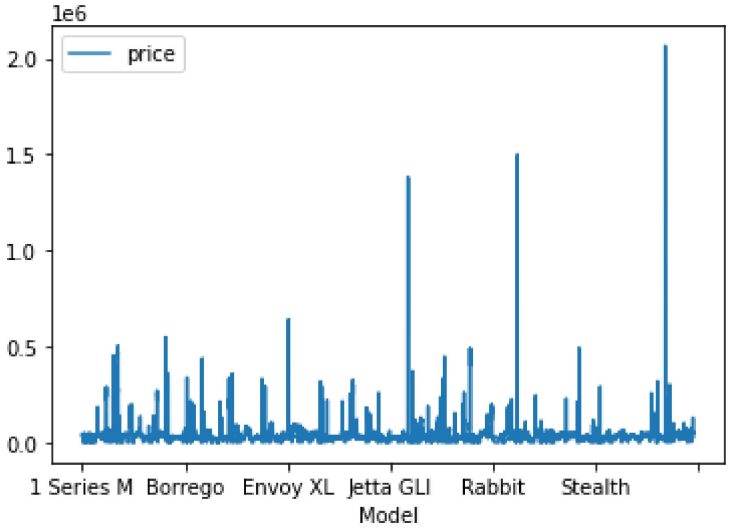

1e6
price
2.0
1.5
1.0
0.5
0.0
0
50
100
150
200
250
300
350
MPG_H

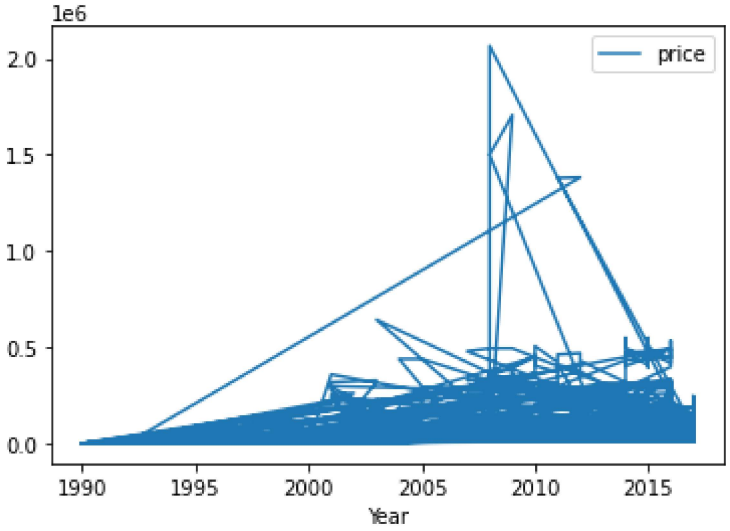

1e6
price
2.0
1.5
1.0
0.5
0.0
20
40
60
80
100
120
140
MPG_c

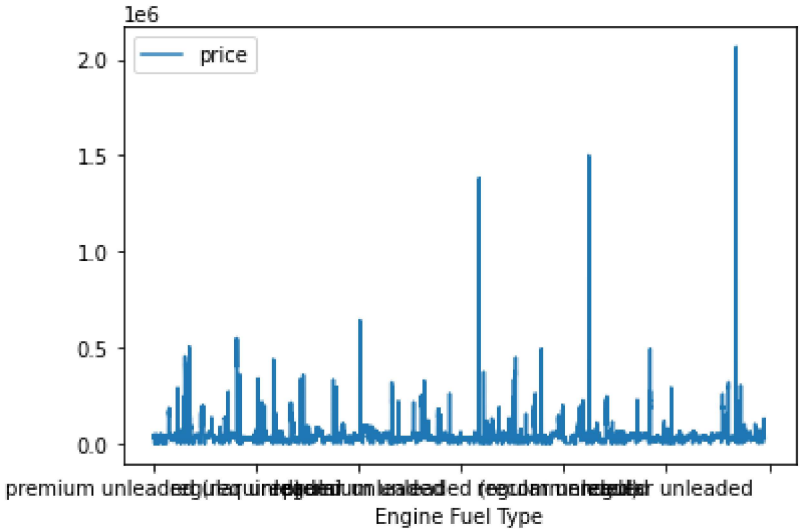

1e6
price
2.0
1.5
1.0
0.5
0.0
0
1000
2000
3000
4000
5000
Popularity

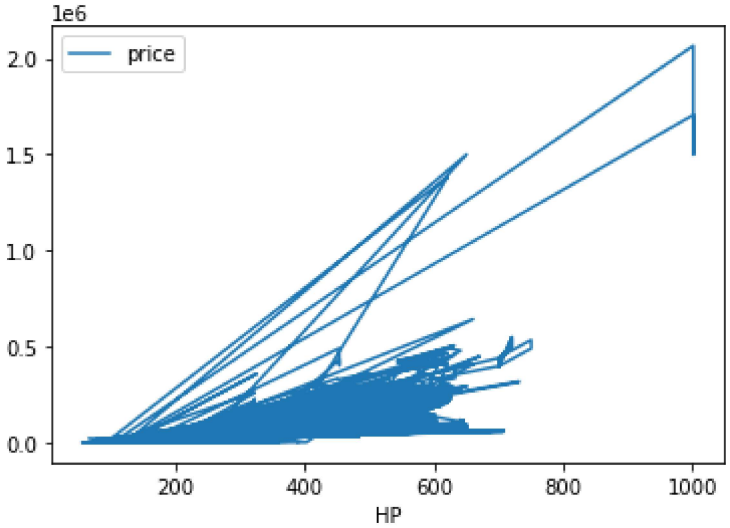
In [109]:
```
for i in car.columns:
    if(i!='price'):
        car.plot(x=i,y='price')
```

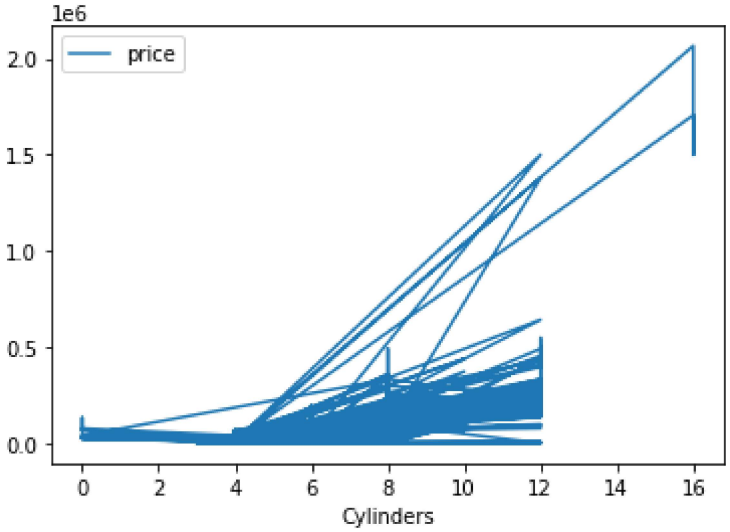

1e6
price
2.0
1.5
1.0
0.5
0.0
BMW
Kia
GMC
Volkswagen
Volkswagen
Dodge
Make

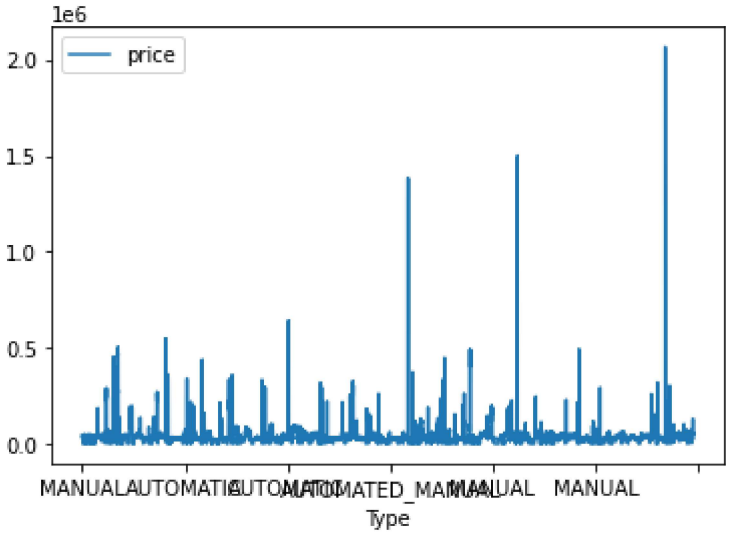

1e6
price
2.0
1.5
1.0
0.5
0.0
1 Series M
Borrego
Envoy XL
Jetta GLI
Rabbit
Stealth
Model

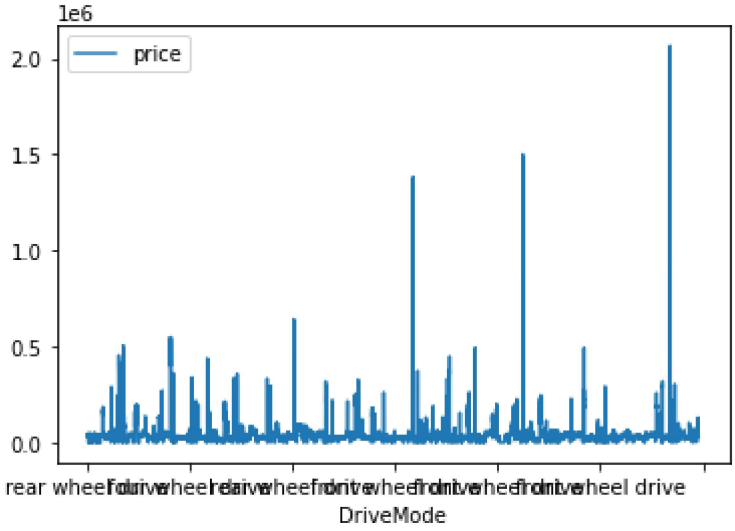

1e6
price
2.0
1.5
1.0
0.5
0.0
1990
1995
2000
2005
2010
2015
Year

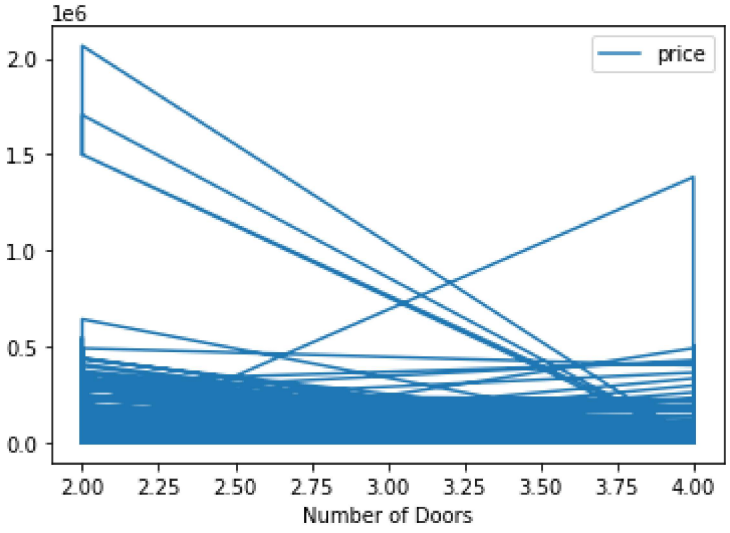

1e6
price
2.0
1.5
1.0
0.5
0.0
premium unleaded
unleaded
Engine Fuel Type

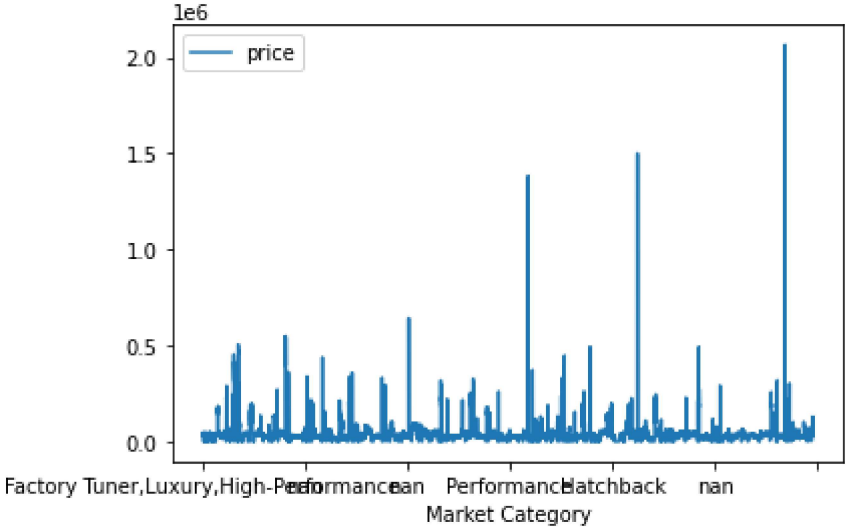

1e6
price
2.0
1.5
1.0
0.5
0.0
200
400
600
800
1000
HP

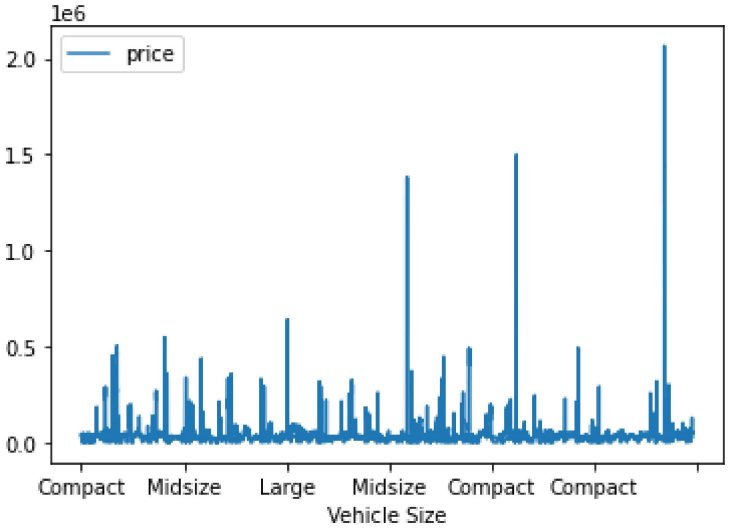

1e6
price
2.0
1.5
1.0
0.5
0.0
0
2
4
6
8
10
12
14
16
Cylinders

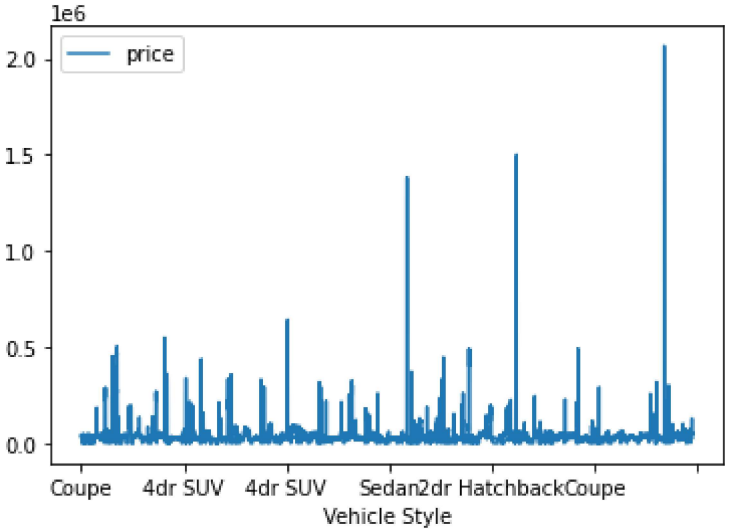

1e6
price
2.0
1.5
1.0
0.5
0.0
MANUALAUTOMATIC
MANUAL
Type

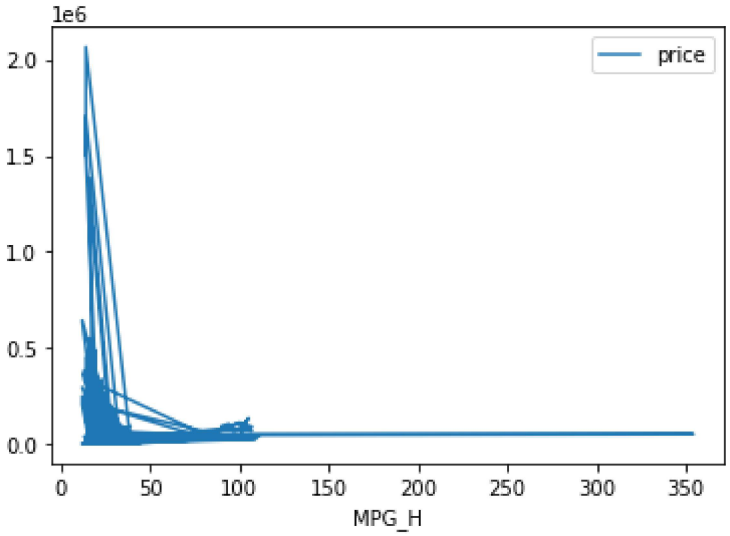

1e6
price
2.0
1.5
1.0
0.5
0.0
DriveMode

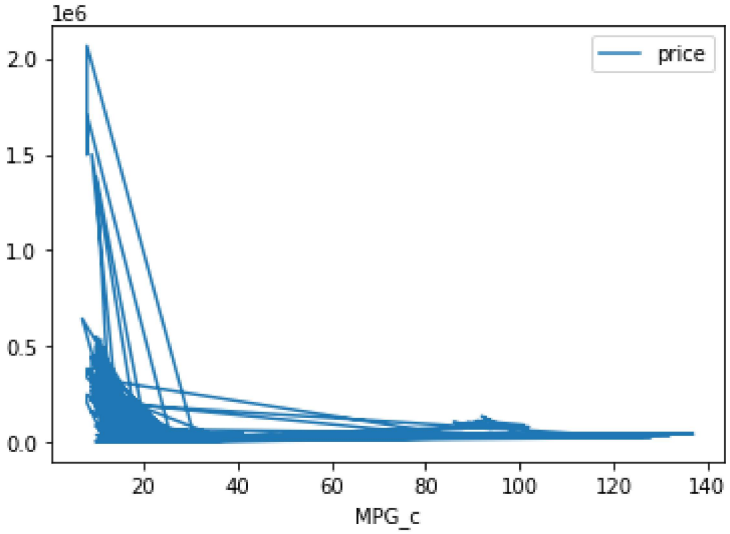

1e6
price
2.0
1.5
1.0
0.5
0.0
2.00
2.25
2.50
2.75
3.00
3.25
3.50
3.75
4.00
Number of Doors

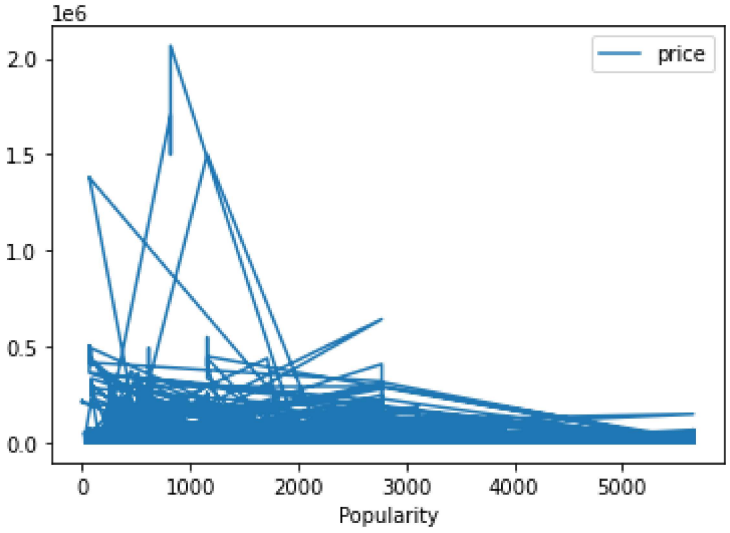

1e6
price
2.0
1.5
1.0
0.5
0.0
Factory Tuner,Luxury,High-Performance
nan
Performance
Hatchback
nan
Market Category


1e6
price
2.0
1.5
1.0
0.5
0.0
Compact
Midsize
Large
Midsize
Compact
Compact
Vehicle Size


1e6
price
2.0
1.5
1.0
0.5
0.0
Coupe
4dr SUV
4dr SUV
Sedan
2dr Hatchback
Coupe
Vehicle Style







Removing some features based on above 2D plots We can see every feature is compared with price, now we are removing **"Make,Model,Engine Fuel Type,Type,Market Category,Vehicle size,Vehicle style"** these features. It doesn't mean that they dont pay role in price variation but not significantly.

In [110]:
```
car=car.drop(['Make', 'Model', 'Engine Fuel Type','Type','Market Category', 'Vehicle Size', 'Vehicle Style'], axis=1)
car.head(5)
```

Out[110]:

	Year	HP	Cylinders	DriveMode	Number of Doors	MPG_H	MPG_c	Popularity	price
0	2011	335.0	6.0	rear wheel drive	2.0	26	19	3916	46135
1	2011	300.0	6.0	rear wheel drive	2.0	28	19	3916	40650
2	2011	300.0	6.0	rear wheel drive	2.0	28	20	3916	36350
3	2011	230.0	6.0	rear wheel drive	2.0	28	18	3916	29450
4	2011	230.0	6.0	rear wheel drive	2.0	28	18	3916	34500

Present Shape of DataFrame car

In [111]:
```
car.shape
```

Out[111]:
```
(11914, 9)
```

In [112]:
```
#Columns
car.columns
```

Out[112]:
```
Index(['Year', 'HP', 'Cylinders', 'DriveMode', 'Number of Doors', 'MPG_H',
       'MPG_c', 'Popularity', 'price'],
      dtype='object')
```

Duplicate rows deletion

In [113]:
```
car[car.duplicated()]
```

Out[113]:

	Year	HP	Cylinders	DriveMode	Number of Doors	MPG_H	MPG_c	Popularity	price
14	2013	230.0	6.0	rear wheel drive	2.0	28	19	3916	31500
18	1992	172.0	6.0	front wheel drive	4.0	24	17	3105	2000
20	1992	172.0	6.0	front wheel drive	4.0	24	17	3105	2000
24	1993	172.0	6.0	front wheel drive	4.0	24	17	3105	2000
25	1993	172.0	6.0	front wheel drive	4.0	24	17	3105	2000
...	...	...	...	...	...	...	...	...	...
11481	1998	95.0	4.0	four wheel drive	2.0	26	22	481	2000
11603	2017	302.0	4.0	all wheel drive	4.0	29	20	870	46350
11604	2017	240.0	4.0	front wheel drive	4.0	30	23	870	40950
11708	2008	252.0	6.0	all wheel drive	4.0	22	15	481	29149
11717	2008	252.0	6.0	front wheel drive	4.0	22	16	481	27499

908 rows × 9 columns

Found total 908 rows which are duplicates

In [114]:
```
car=car.drop_duplicates()
print("after removing duplicates num of rows and columns",car.shape)
```

```
after removing duplicates num of rows and columns (11006, 9)
```

Now we need to remove null values

In [115]:
```
#checking for existance of null values
car.isnull().values.any()
```

Out[115]: True

In [116]:
```
#No of null values in entire dataframe
car.isnull().sum().sum()
```

Out[116]: 105

In [117]:
```
#droping null values from dataframe
print("Data frame shape before droping nulls",car.shape)
car=car.dropna()
print("Data frame shape after droping nulls",car.shape)
```

```
Data frame shape before droping nulls (11006, 9)
Data frame shape after droping nulls (10907, 9)
```

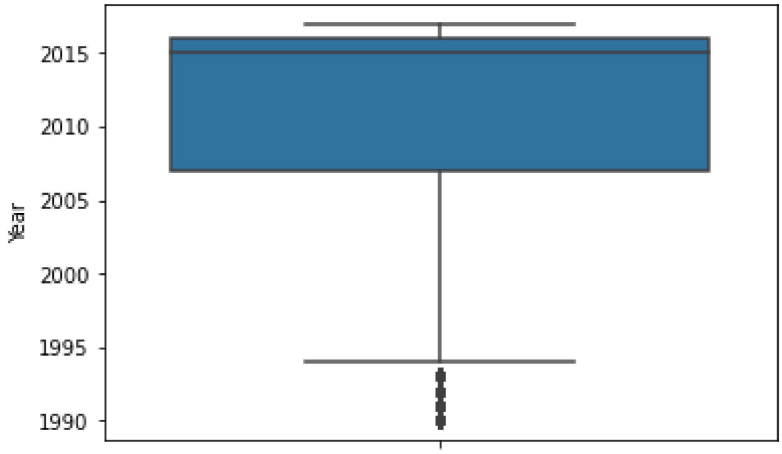
Removing the Outliers

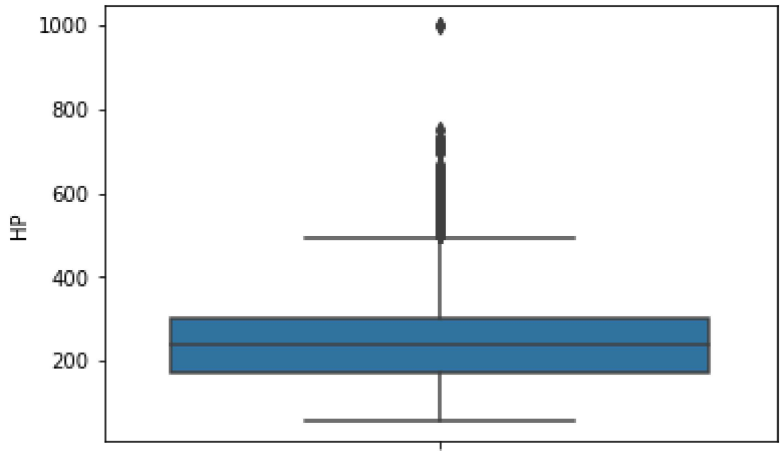
We can detect outliers using boxplots,Zscore,percentiles etc. But here we proceed with boxplots, if it cause difficulty in detection outliers then we move into other methods of finding outliers.

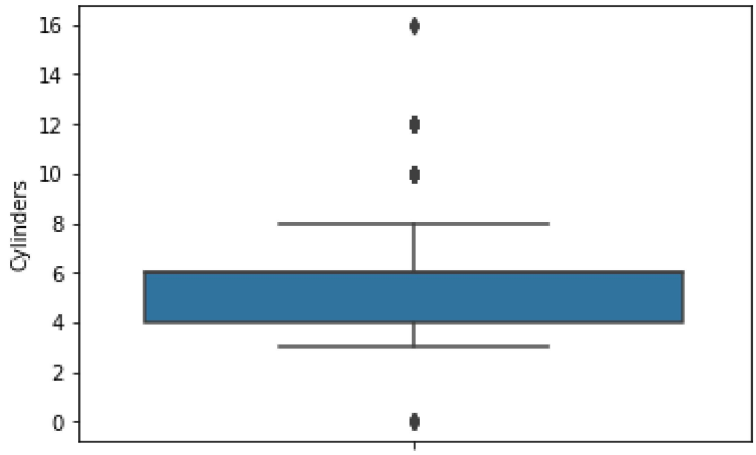
In [118]:
```
car.columns
```

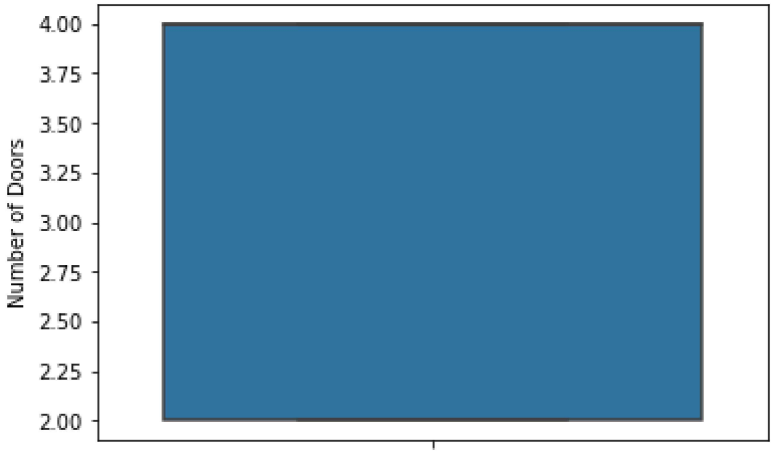
Out[118]:
```
Index(['Year', 'HP', 'Cylinders', 'DriveMode', 'Number of Doors', 'MPG_H',
       'MPG_c', 'Popularity', 'price'],
      dtype='object')
```

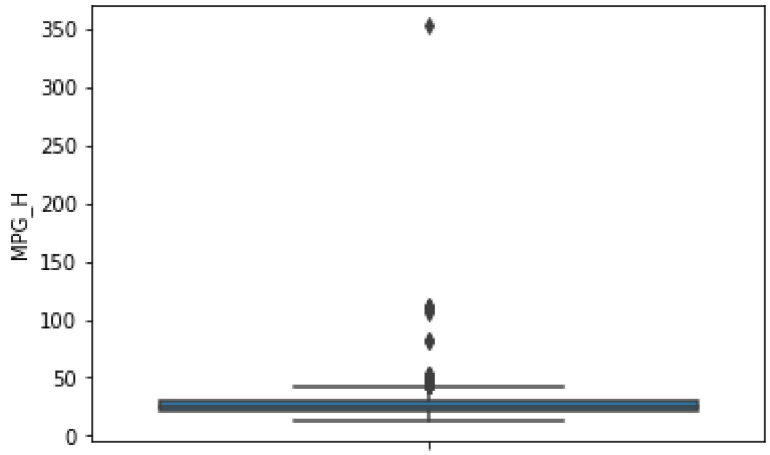
In [119]:
```
for i in car.columns:
    if (i !='DriveMode'):
        sns.boxplot(y=i,data=car)
        mp.pyplot.show()
    #sns.boxplot(y=car[i])
#sns.boxplot(y=i,data=car)
```

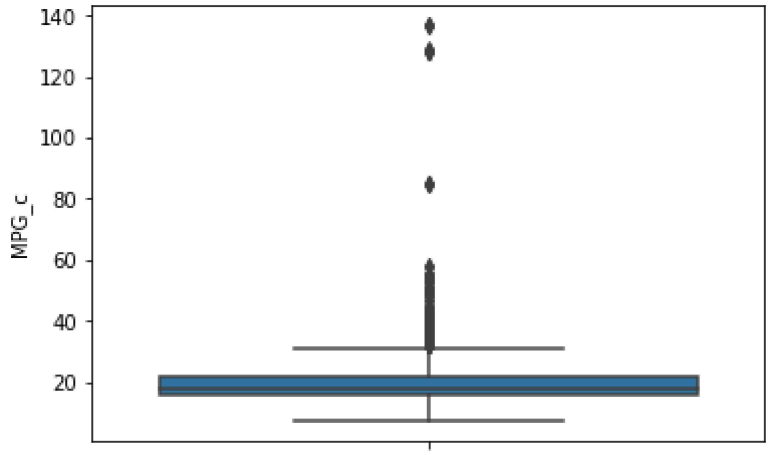

Year
2015
2010
2005
2000
1995
1990

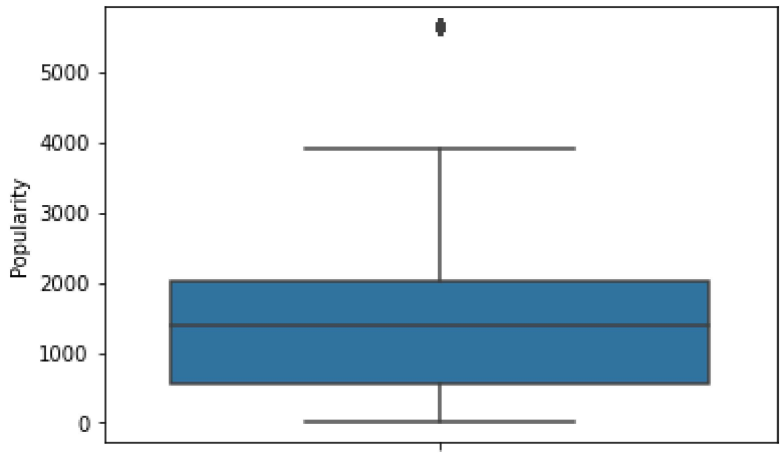

HP
1000
800
600
400
200

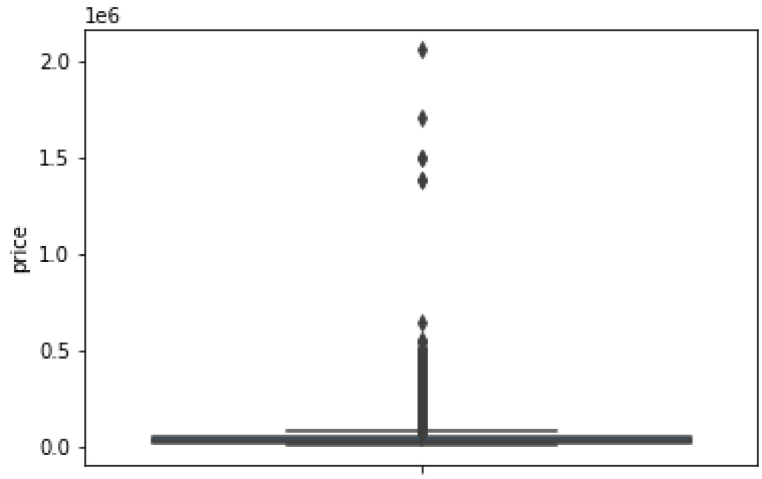

Cylinders
16
14
12
10
8
6
4
2
0


Number of Doors
4.00
3.75
3.50
3.25
3.00
2.75
2.50
2.25
2.00


MPG_H
350
300
250
200
150
100
50
0


MPG_c
140
120
100
80
60
40
20





In [120]:
```
#sns.boxplot(y=car['price'],x=car['DriveMode'])
```

In [121]:
```
car.columns
```

Out[121]:
```
Index(['Year', 'HP', 'Cylinders', 'DriveMode', 'Number of Doors', 'MPG_H',
       'MPG_c', 'Popularity', 'price'],
      dtype='object')
```

From the above boxplots its

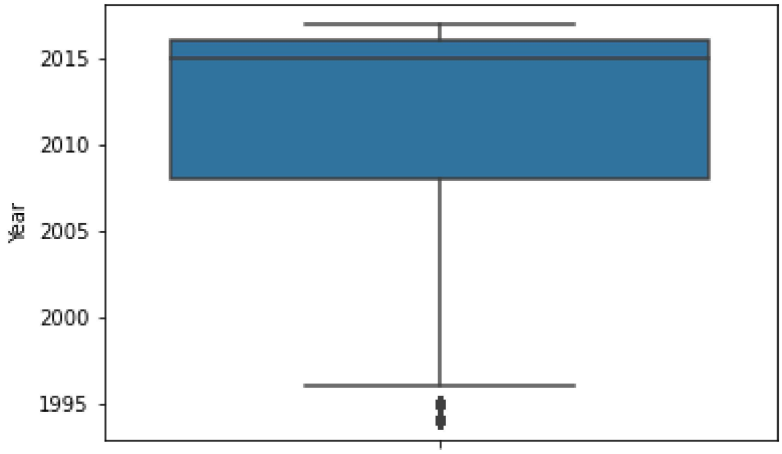
In [122]:
```
for i in car.columns:
    if (i!='DriveMode'):
        Q1 = car[i].quantile(0.25)
        Q3 = car[i].quantile(0.75)
        IQR = Q3 - Q1
        #car = car[~((car[i] < (Q1 - 1.5 * IQR)) |(car[i] > (Q3 + 1.5 * IQ
R)))][i]
        filters = (car[i] >= Q1 - 1.5 * IQR) & (car[i] <= Q3 + 1.5 *IQR)
        car=car.loc[filters]
```

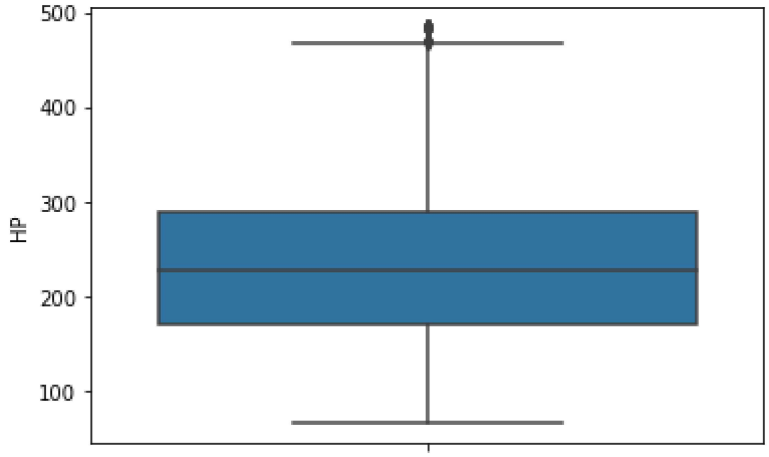
In [123]:
```
print("after removal of outliers from car dataFrame shape of dataframe : ",car
.shape)
```

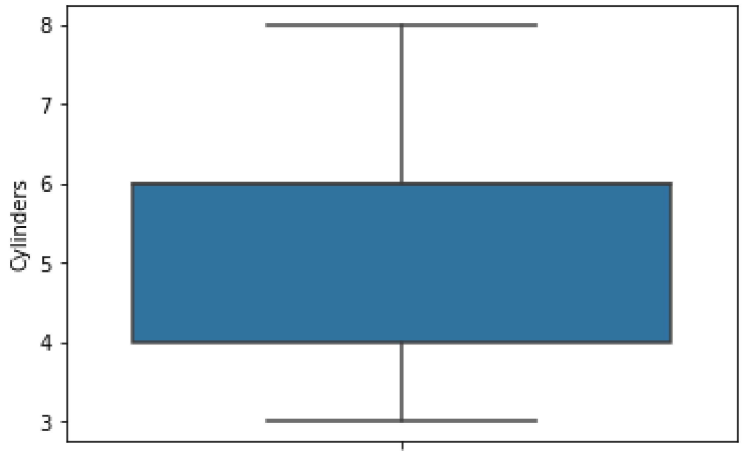
```
after removal of outliers from car dataFrame shape of dataframe :  (8450, 9)
```

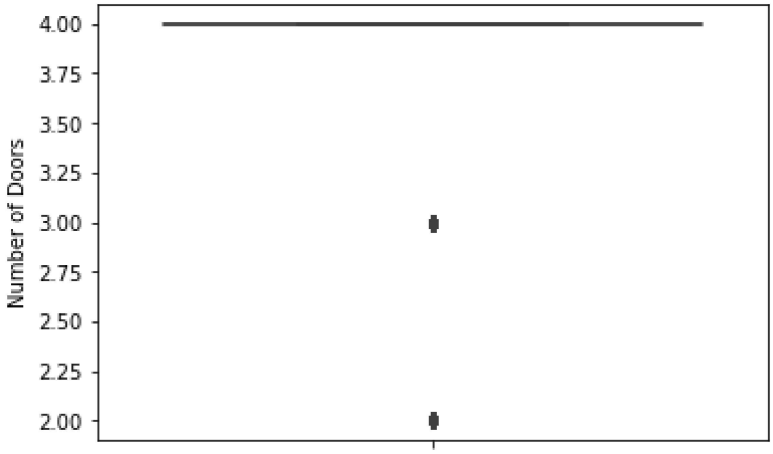
Now ploting the boxplots inorder to check wheather outliers are removed or not.

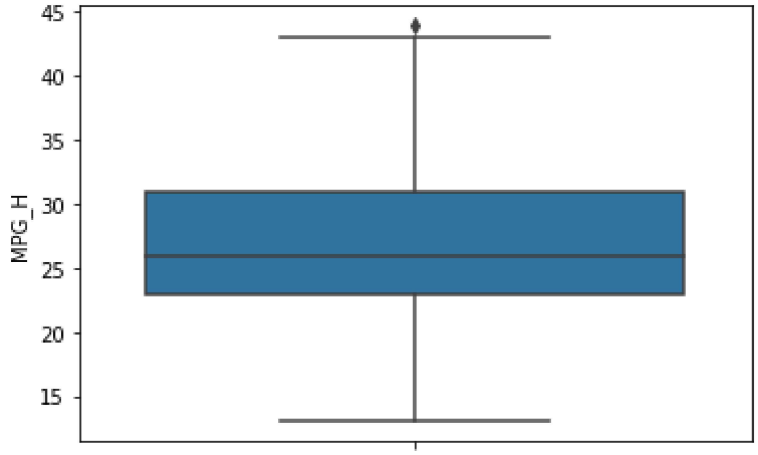
In [124]:
```
for i in car.columns:
    if (i !='DriveMode'):
        sns.boxplot(y=i,data=car)
        mp.pyplot.show()
    #sns.boxplot(y=car[i])
#sns.boxplot(y=i,data=car)
```

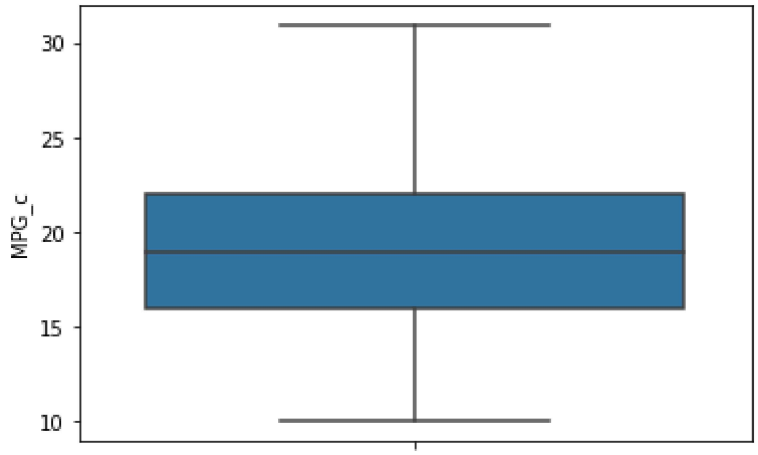

Year
2015
2010
2005
2000
1995

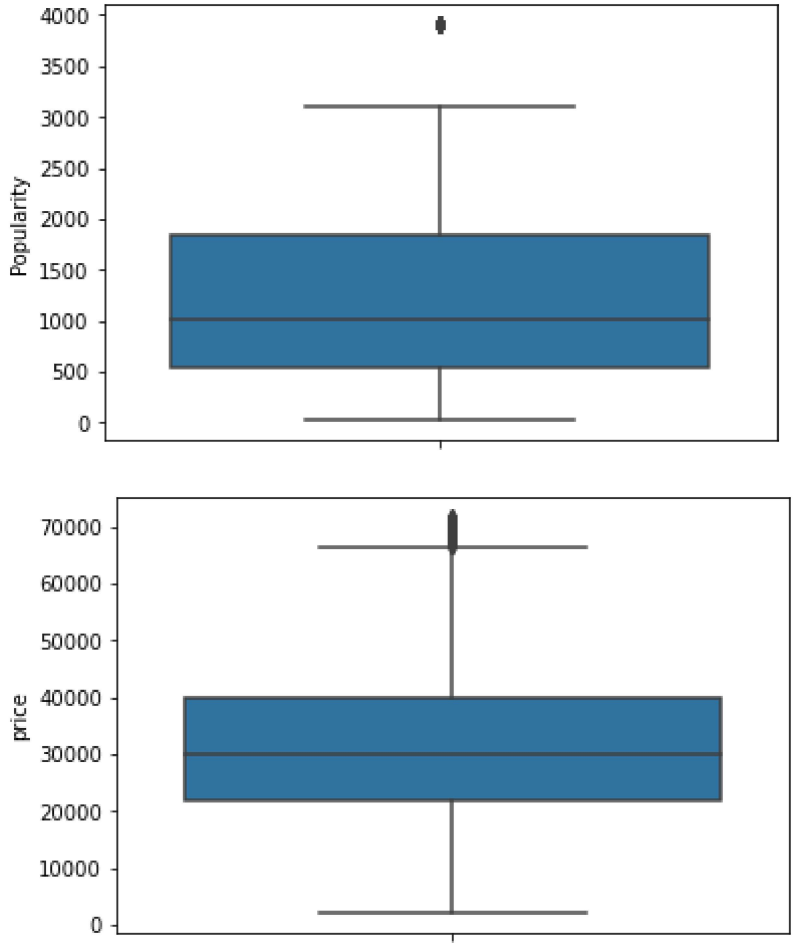

HP
500
400
300
200
100


Cylinders
8
7
6
5
4
3


Number of Doors
4.00
3.75
3.50
3.25
3.00
2.75
2.50
2.25
2.00


MPG_H
45
40
35
30
25
20
15

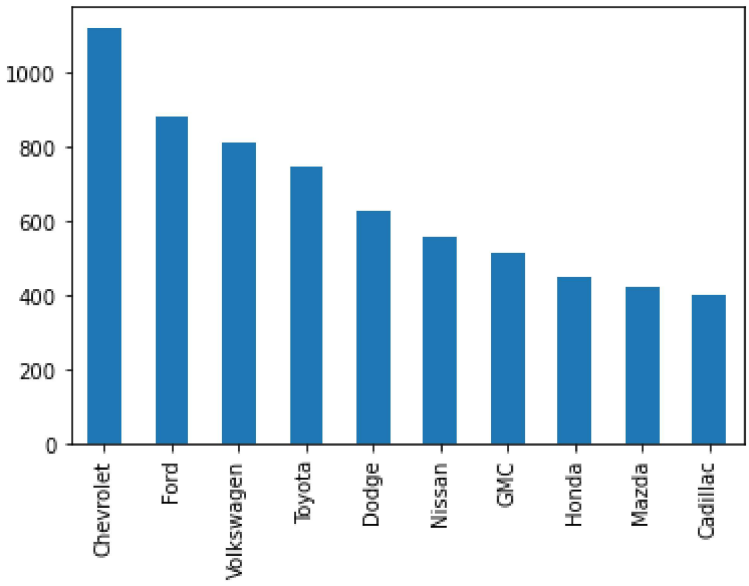

MPG_c
30
25
20
15
10



Top 10 represented car makers with there mean price

In [125]:
```
#before eliminating make column we saved that in Makers as a series object
Makers.head(10).plot(kind="bar")
```

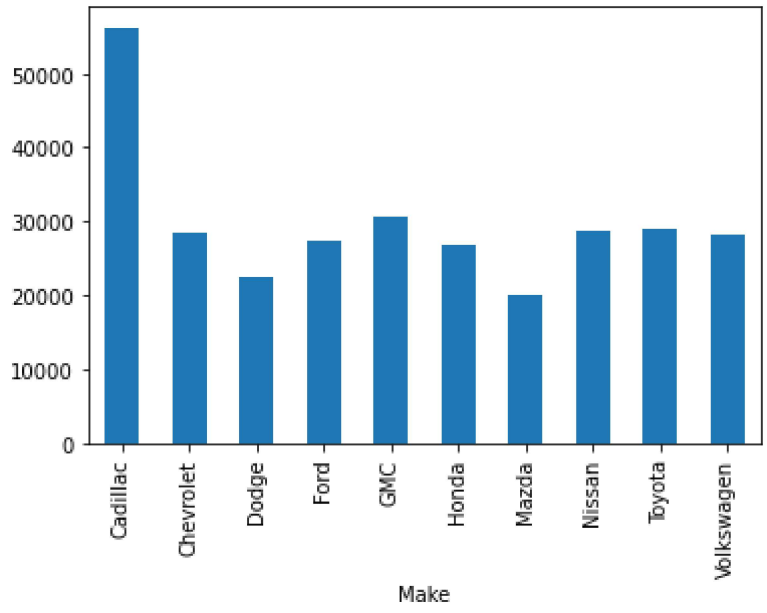
Out[125]: <matplotlib.axes._subplots.AxesSubplot at 0x1ce23727d30>



In [126]:
```
MeanAmount = MP[['Make','price']].loc[(MP['Make'] == 'Chevrolet')|
                (MP['Make'] == 'Ford')|
                (MP['Make'] == 'Volkswagen')|
                (MP['Make'] == 'Toyota')|
                (MP['Make'] == 'Dodge')|
                (MP['Make'] == 'Nissan')|
                (MP['Make'] == 'GMC')|
                (MP['Make'] == 'Honda')|
                (MP['Make'] == 'Mazda')|(MP['Make']=='Cadillac')].groupby('Mak
e').mean()
print(round(MeanAmount['price'],2))
round(MeanAmount['price'],2).plot(kind="bar")
#price rounding off to 2 decimals
```

```
Make
Cadillac      56231.32
Chevrolet     28350.39
Dodge         22390.06
Ford          27399.27
GMC           30493.30
Honda         26674.34
Mazda         20039.38
Nissan        28583.43
Toyota        29030.02
Volkswagen    28102.38
Name: price, dtype: float64
```

Out[126]: <matplotlib.axes._subplots.AxesSubplot at 0x1ce237898b0>



Correlation Matrix

In [127]:
```
car.corr()
```

Out[127]:

	Year	HP	Cylinders	Number of Doors	MPG_H	MPG_c	Popularity	price
Year	1.000000	0.318256	-0.112402	0.244520	0.375230	0.337014	0.197737	0.588409
HP	0.318256	1.000000	0.735256	0.040560	-0.457774	-0.557828	0.086682	0.740093
Cylinders	-0.112402	0.735256	1.000000	0.019035	-0.693411	-0.747326	0.008734	0.384355
Number of Doors	0.244520	0.040560	0.019035	1.000000	0.011900	0.027839	-0.075513	0.135580
MPG_H	0.375230	-0.457774	-0.693411	0.011900	1.000000	0.937943	0.097126	-0.120640
MPG_c	0.337014	-0.557828	-0.747326	0.027839	0.937943	1.000000	0.085912	-0.196735
Popularity	0.197737	0.086682	0.008734	-0.075513	0.097126	0.085912	1.000000	0.105092
price	0.588409	0.740093	0.384355	0.135580	-0.120640	-0.196735	0.105092	1.000000

conclusions drawn from correlation matrix

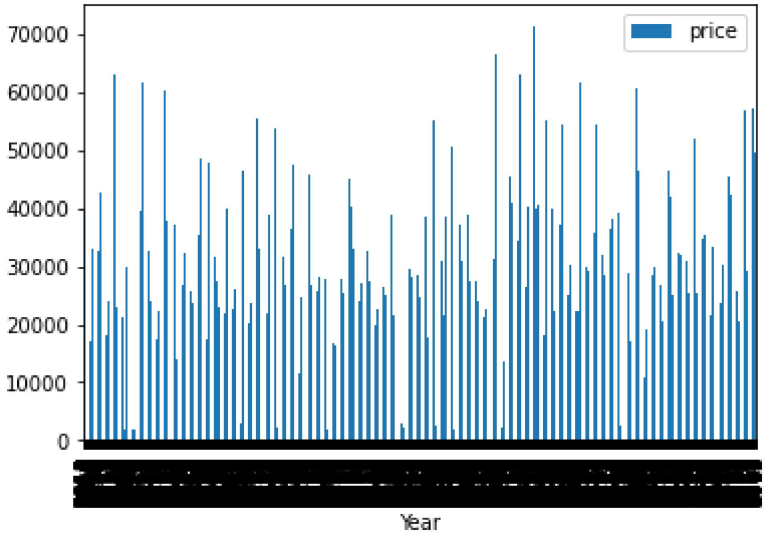
1) HP and Cylinders are +vely correlated.
2) HP and Price are +vely correlated.
3) Cylinders and MPG_H are -vely correlated.
4) Cylinders and MPG_c are -vely correlated.
5) MPG_H and MPG_c are +vely correlated.

Exploratory Data analysis

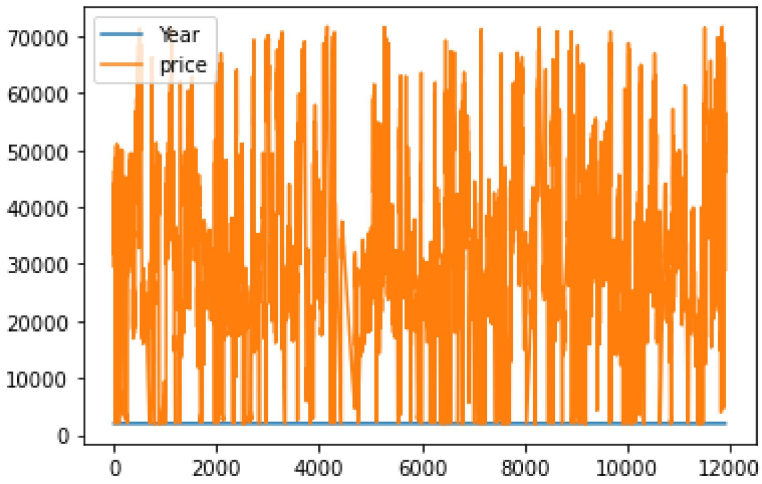
In [128]:
```
mp.pyplot.figure(figsize=(20,8))
car.plot.bar(x="Year", y="price")
```

Out[128]: <matplotlib.axes._subplots.AxesSubplot at 0x1ce2378ee80>

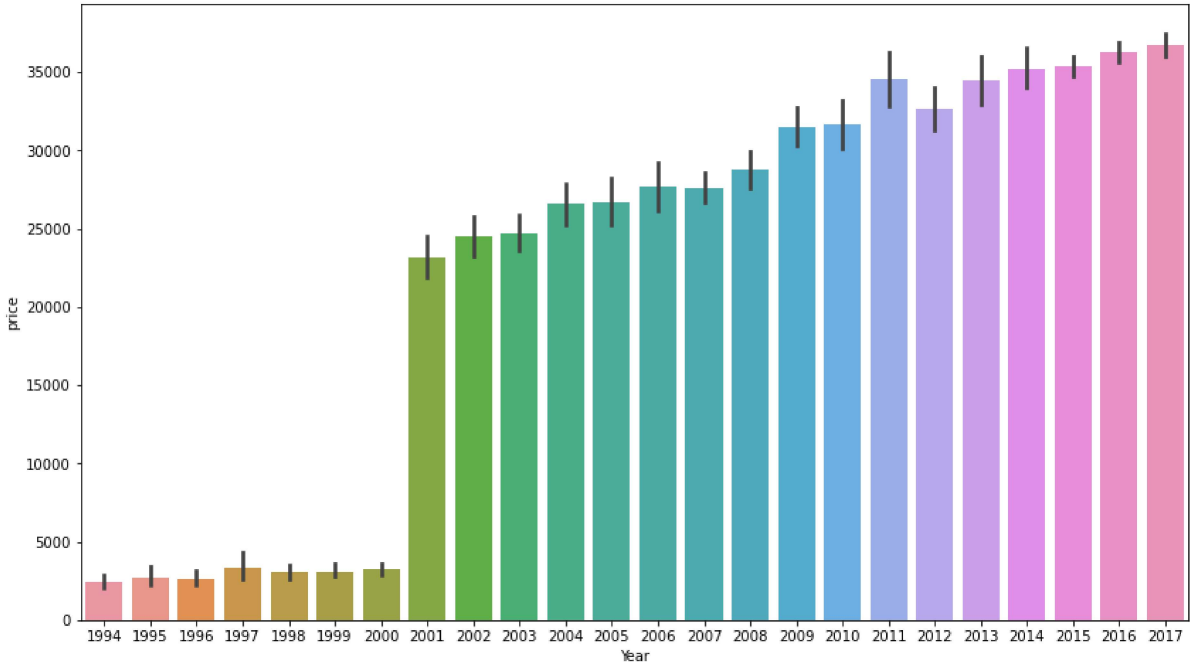
<Figure size 1440x576 with 0 Axes>

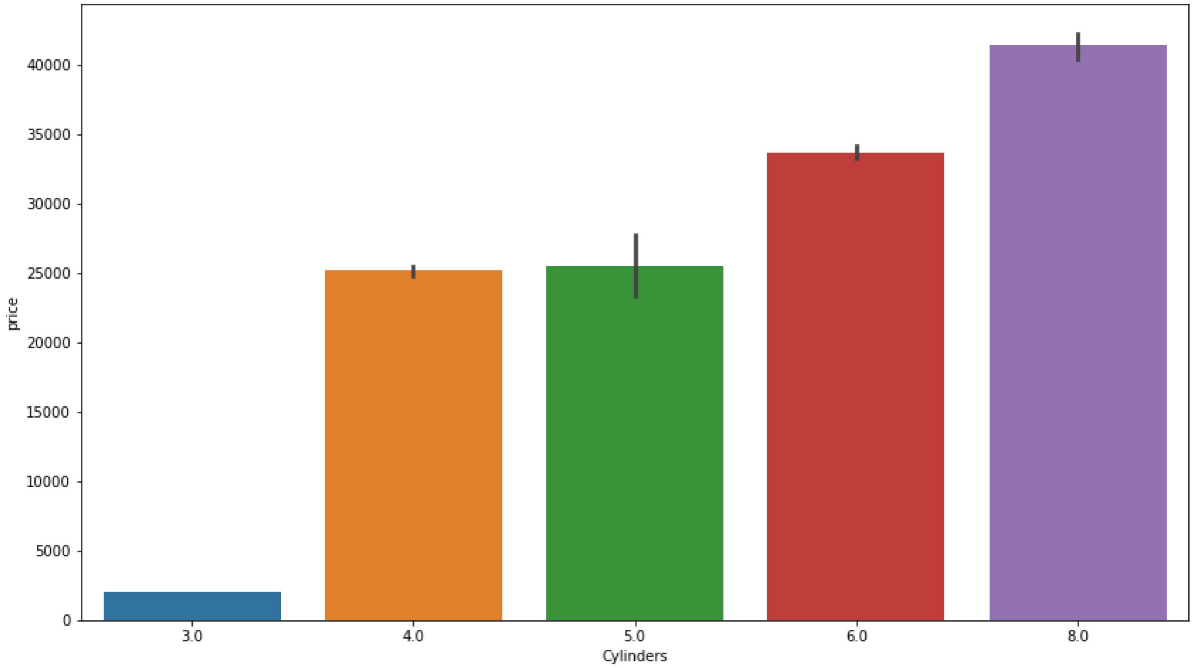


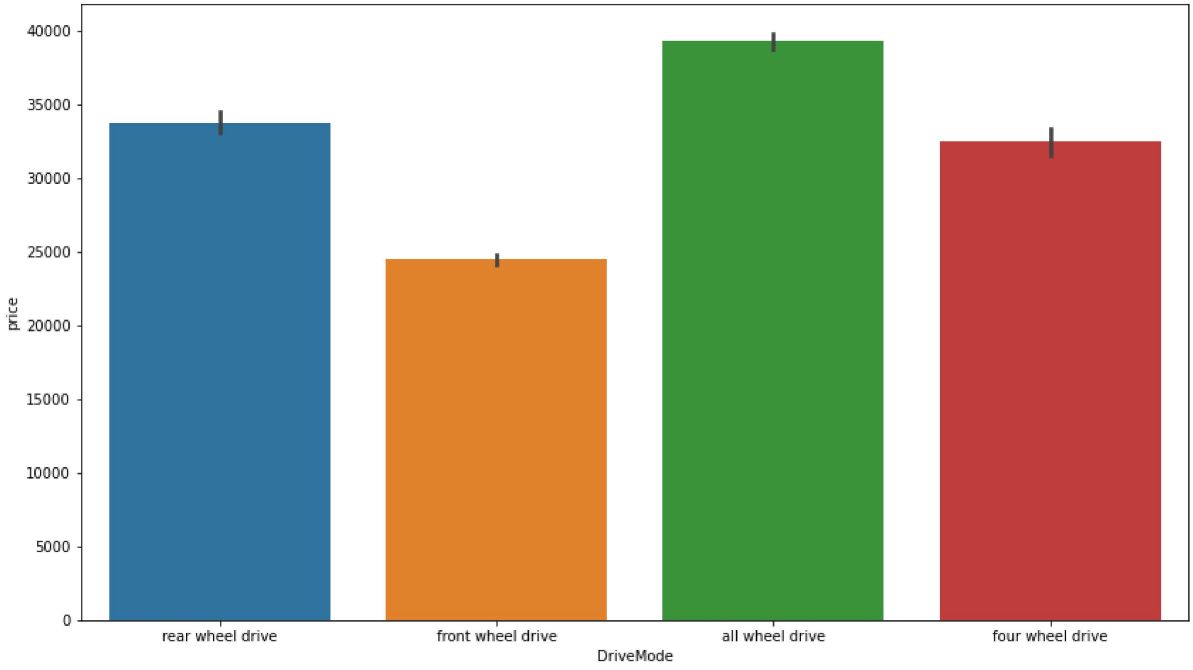
In [129]:
```
car[['Year','price']].plot()
mp.pyplot.show()
```

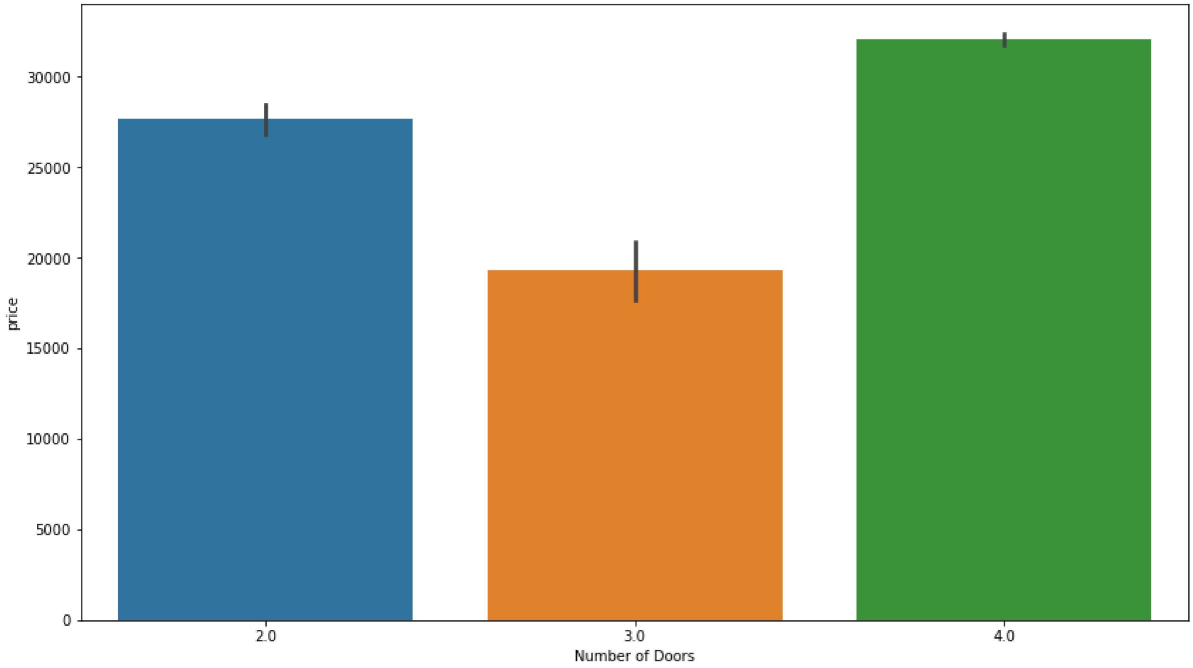


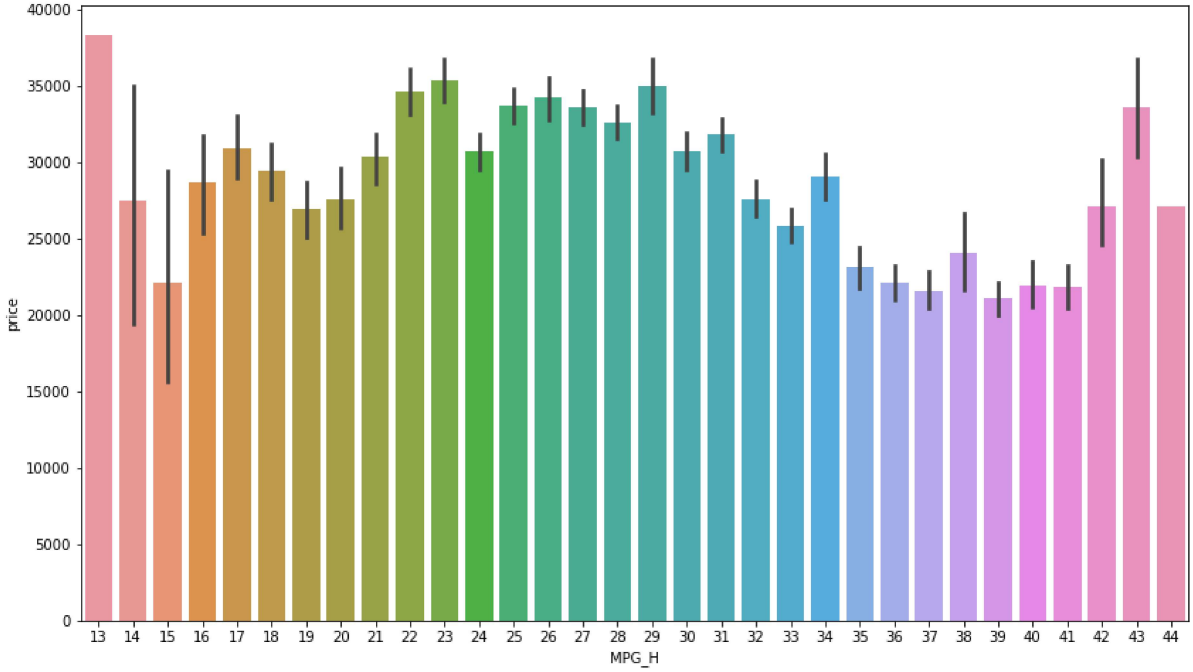
In [130]:
```
#from above its not clear so we use bars
for i in car.columns:
    if(i!='price' and i!='HP'):
        mp.pyplot.figure(figsize=(14,8))
        sns.barplot(car[i],car['price'])
        mp.pyplot.show()
```

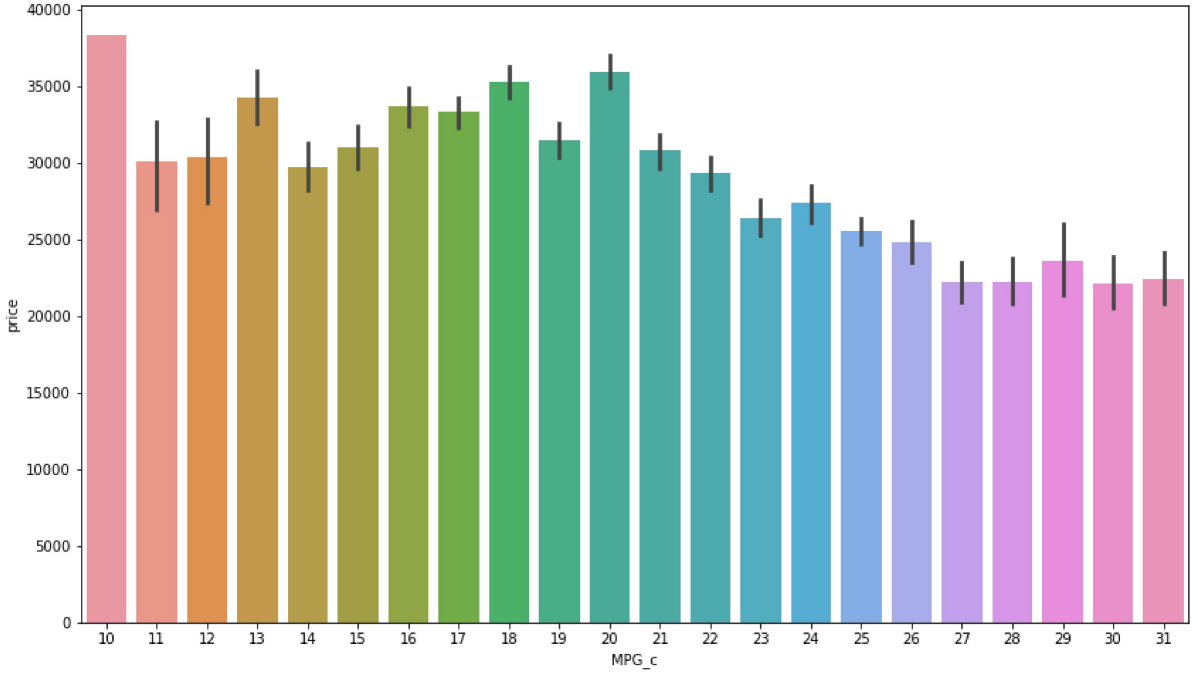

price
Year
0
5000
10000
15000
20000
25000
30000
35000
1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017

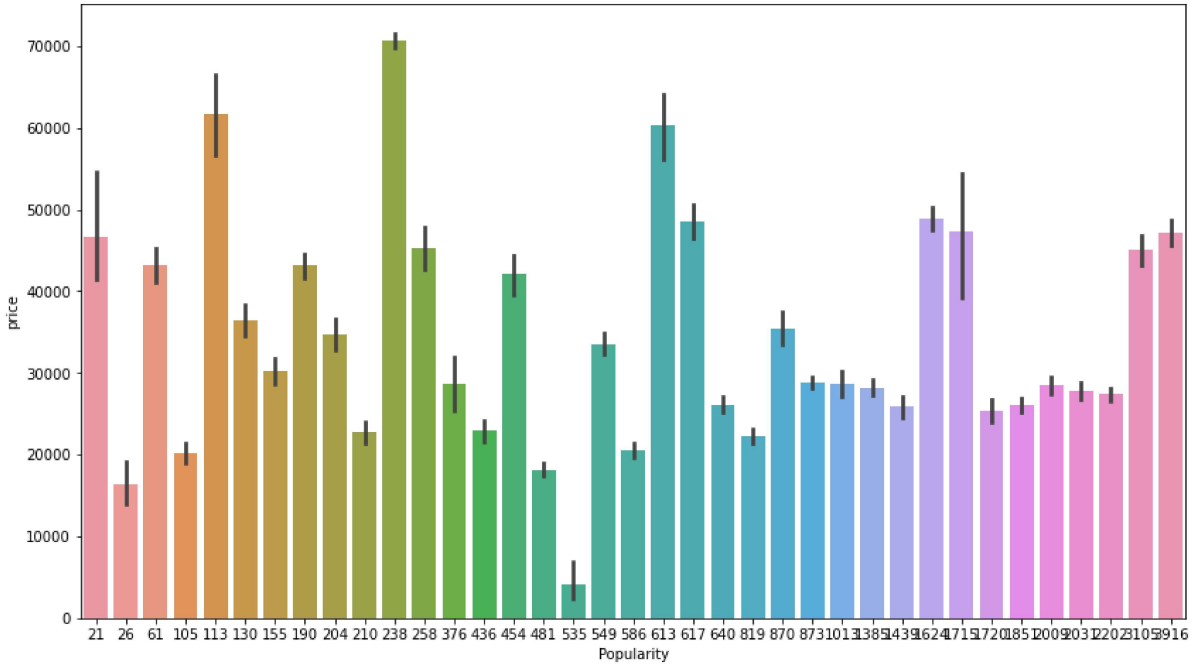

price
Cylinders
0
5000
10000
15000
20000
25000
30000
35000
40000
3.0
4.0
5.0
6.0
8.0


40000
35000
30000
25000
20000
15000
10000
5000
0
price
rear wheel drive
front wheel drive
all wheel drive
four wheel drive
DriveMode


30000
25000
20000
15000
10000
5000
0
price
2.0
3.0
4.0
Number of Doors


price
MPG_H


price
MPG_c



Conclusions:

1) From 2001 price got jumped from below 5k to above 20k
2) Less price for 3 cylinders car and almost same cost for 4,5 where as 6,7 are more price than all of them
3) Front wheel drive mode is less priced among others
4) Cars with 2,4 doors are most priced than 3 doored
5) Almost same priced for every MPG_H and MPG_C
6) Popularity not effecting the price effectively

In [ ]: