Plotting with Matplotlib & Seaborn

In [1]:

```
import pandas as pd
import numpy as np
import warnings
import matplotlib.pyplot as plt
import seaborn as sns
warnings.filterwarnings('ignore')
```

In [2]:

```
data = pd.read_csv('./Car_sales.csv')
data.head(10)
```

Out[2]:

	Manufacturer	Model	Sales_in_thousands	__year_resale_value	Vehicle_type	Price_in_thousands	E
0	Acura	Integra	16.919	16.360	Passenger	21.50	
1	Acura	TL	39.384	19.875	Passenger	28.40	
2	Acura	CL	14.114	18.225	Passenger	NaN	
3	Acura	RL	8.588	29.725	Passenger	42.00	
4	Audi	A4	20.397	22.255	Passenger	23.99	
5	Audi	A6	18.780	23.555	Passenger	33.95	
6	Audi	A8	1.380	39.000	Passenger	62.00	
7	BMW	323i	19.747	NaN	Passenger	26.99	
8	BMW	328i	9.231	28.675	Passenger	33.40	
9	BMW	528i	17.527	36.125	Passenger	38.90	

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 157 entries, 0 to 156
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Manufacturer         157 non-null    object 
 1   Model                157 non-null    object 
 2   Sales_in_thousands   157 non-null    float64
 3   __year_resale_value  121 non-null    float64
 4   Vehicle_type         157 non-null    object 
 5   Price_in_thousands   155 non-null    float64
 6   Engine_size          156 non-null    float64
 7   Horsepower           156 non-null    float64
 8   Wheelbase            156 non-null    float64
 9   Width                156 non-null    float64
 10  Length               156 non-null    float64
 11  Curb_weight          155 non-null    float64
 12  Fuel_capacity        156 non-null    float64
 13  Fuel_efficiency      154 non-null    float64
 14  Latest_Launch        157 non-null    object 
 15  Power_perf_factor    155 non-null    float64
dtypes: float64(12), object(4)
memory usage: 19.8+ KB
```

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

Out[4]:

	Sales_in_thousands	__year_resale_value	Price_in_thousands	Engine_size	Horsepower	Wheell
count	157.000000	121.000000	155.000000	156.000000	156.000000	156.00
mean	52.998076	18.072975	27.390755	3.060897	185.948718	107.48
std	68.029422	11.453384	14.351653	1.044653	56.700321	7.64
min	0.110000	5.160000	9.235000	1.000000	55.000000	92.60
25%	14.114000	11.260000	18.017500	2.300000	149.500000	103.00
50%	29.450000	14.180000	22.799000	3.000000	177.500000	107.00
75%	67.956000	19.875000	31.947500	3.575000	215.000000	112.20
max	540.561000	67.550000	85.500000	8.000000	450.000000	138.70

In [5]:
```
data.isnull().mean()*100
```

Out[5]:
```
Manufacturer            0.000000
Model                   0.000000
Sales_in_thousands      0.000000
__year_resale_value    22.929936
Vehicle_type            0.000000
Price_in_thousands      1.273885
Engine_size             0.636943
Horsepower              0.636943
Wheelbase               0.636943
Width                   0.636943
Length                  0.636943
Curb_weight             1.273885
Fuel_capacity           0.636943
Fuel_efficiency         1.910828
Latest_Launch           0.000000
Power_perf_factor       1.273885
dtype: float64
```

In [6]:
```
data.dropna(inplace=True)
data.drop_duplicates(inplace=True)
```

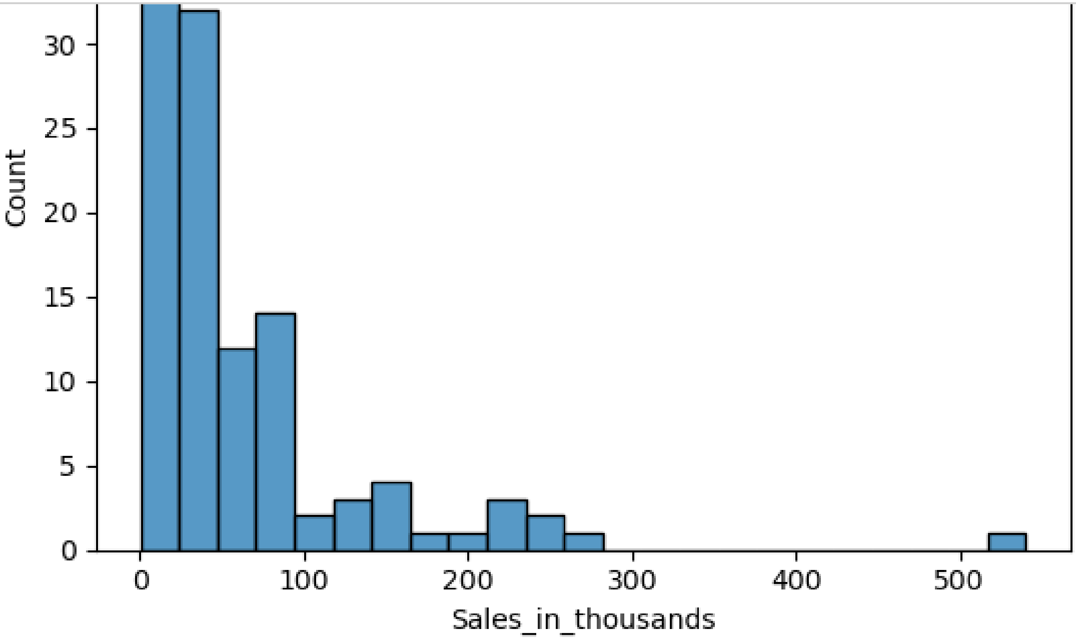
In [7]:
```
data.shape
```

Out[7]:
```
(117, 16)
```

Univariate Analysis

Histogram

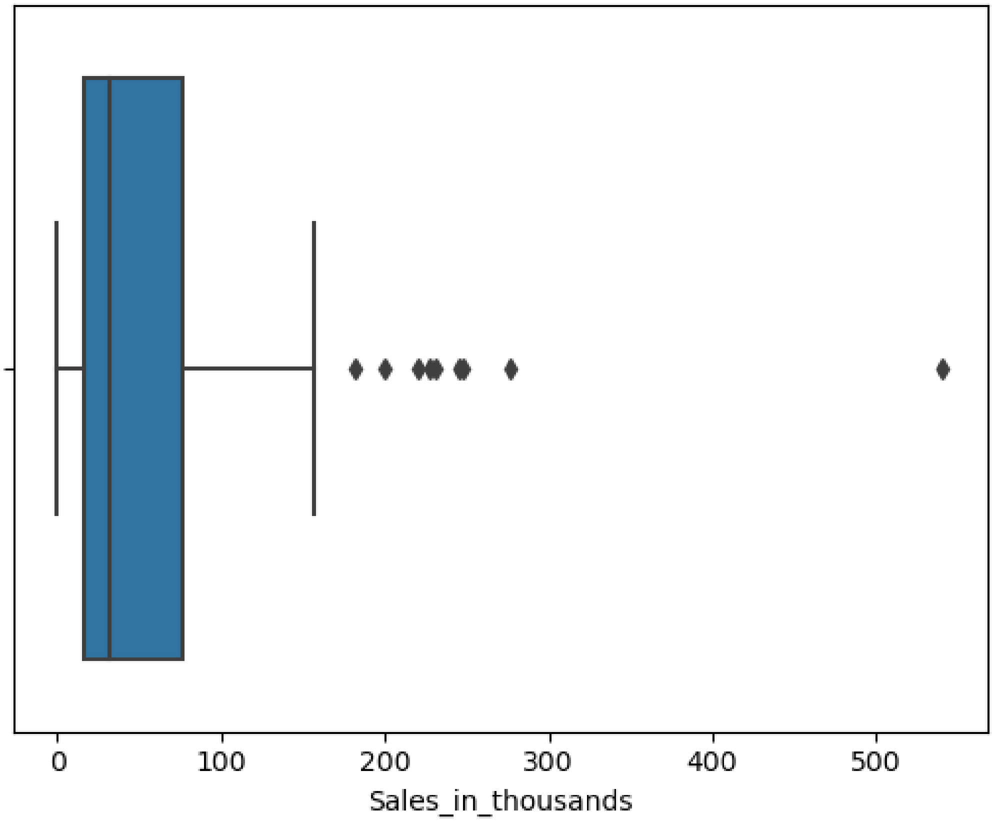
In [8]:
```
sns.histplot(data=data, x='Sales_in_thousands')
plt.show()
```



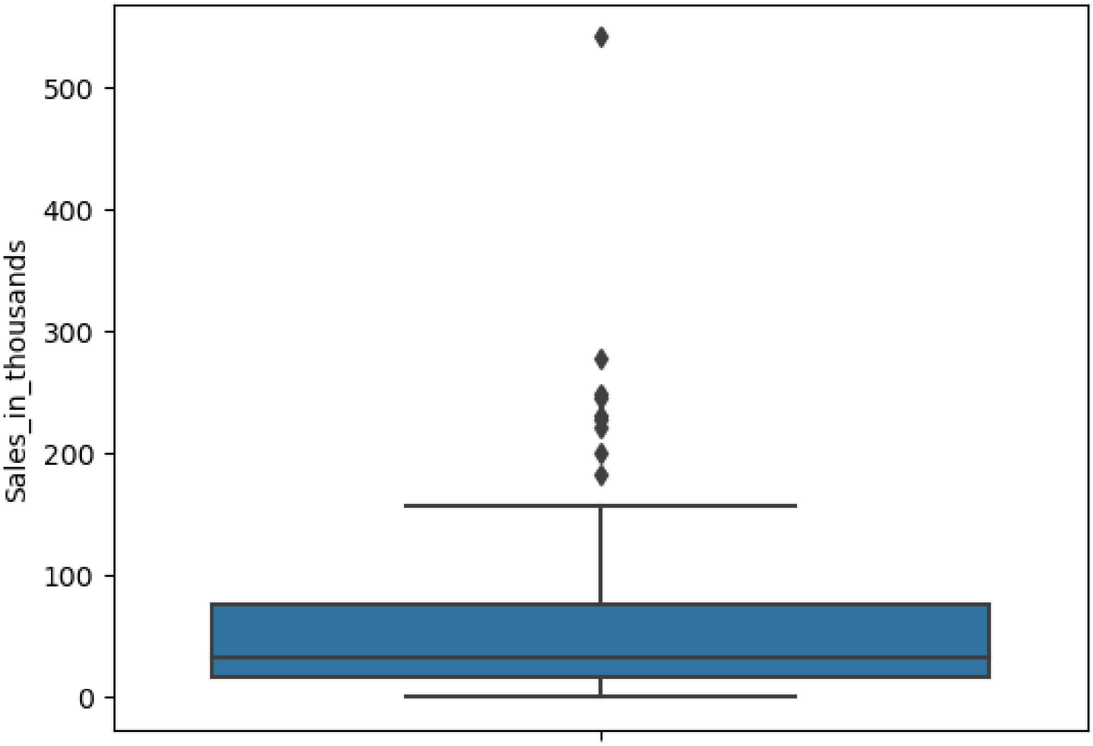
Box plot

In [9]:

```
sns.boxplot(data=data, x='Sales_in_thousands')
plt.show()
```

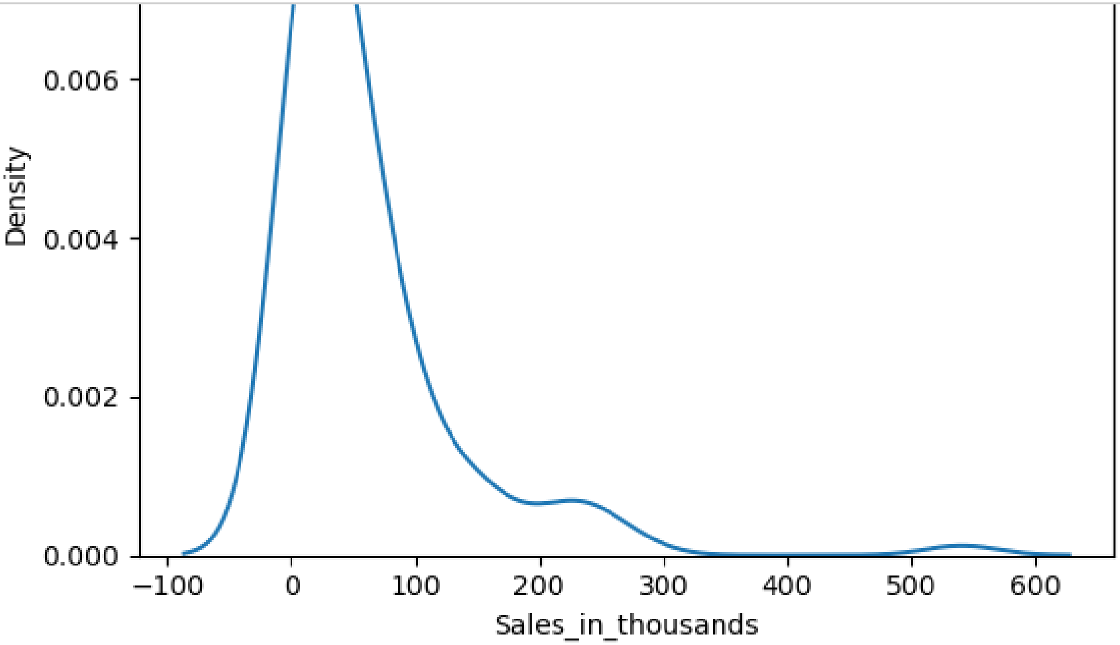


In [10]:
```
sns.boxplot(data=data, y='Sales_in_thousands')
plt.show()
```



Kernel Density Estimation Plot (KDE PLOT)

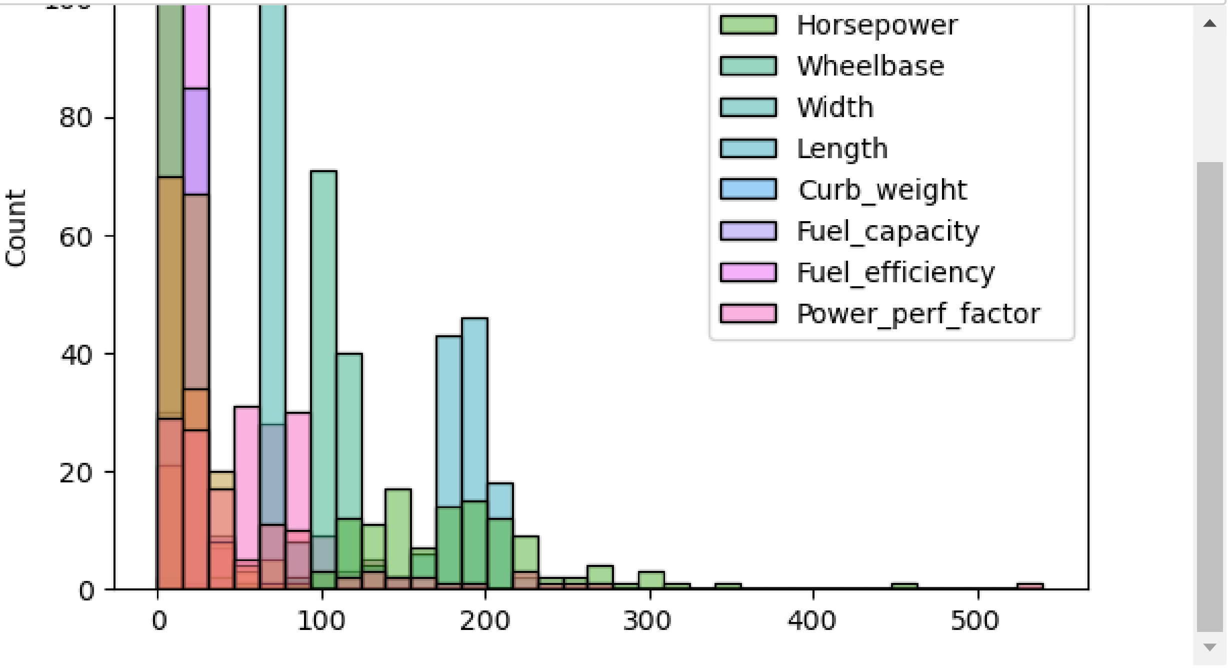
In [11]:
```
sns.kdeplot(data=data.Sales_in_thousands)
plt.show()
```



Sub plots

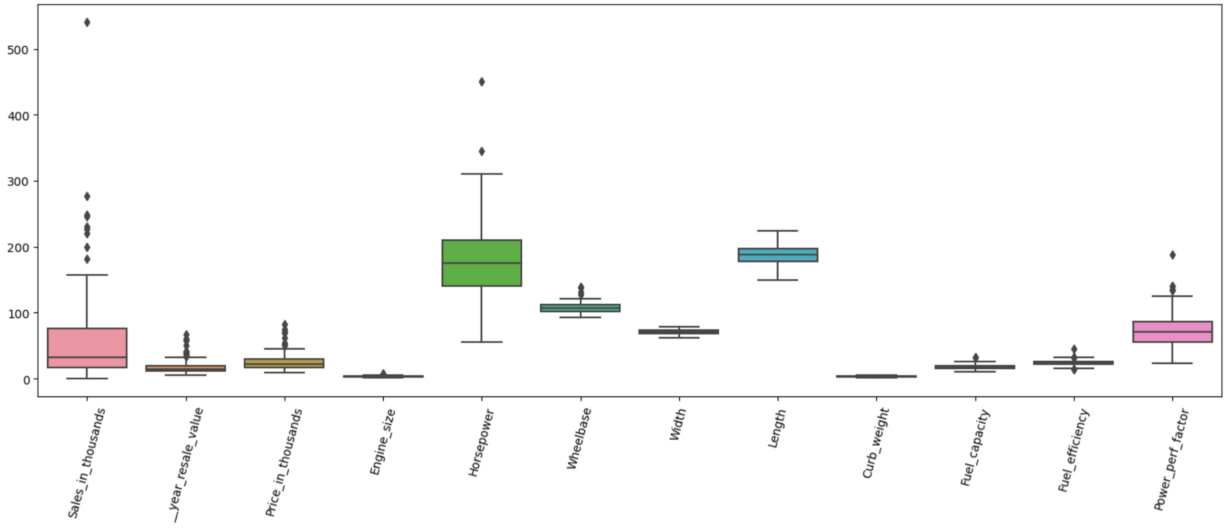
In [12]:

```
sns.histplot(data=data)
plt.show()
```



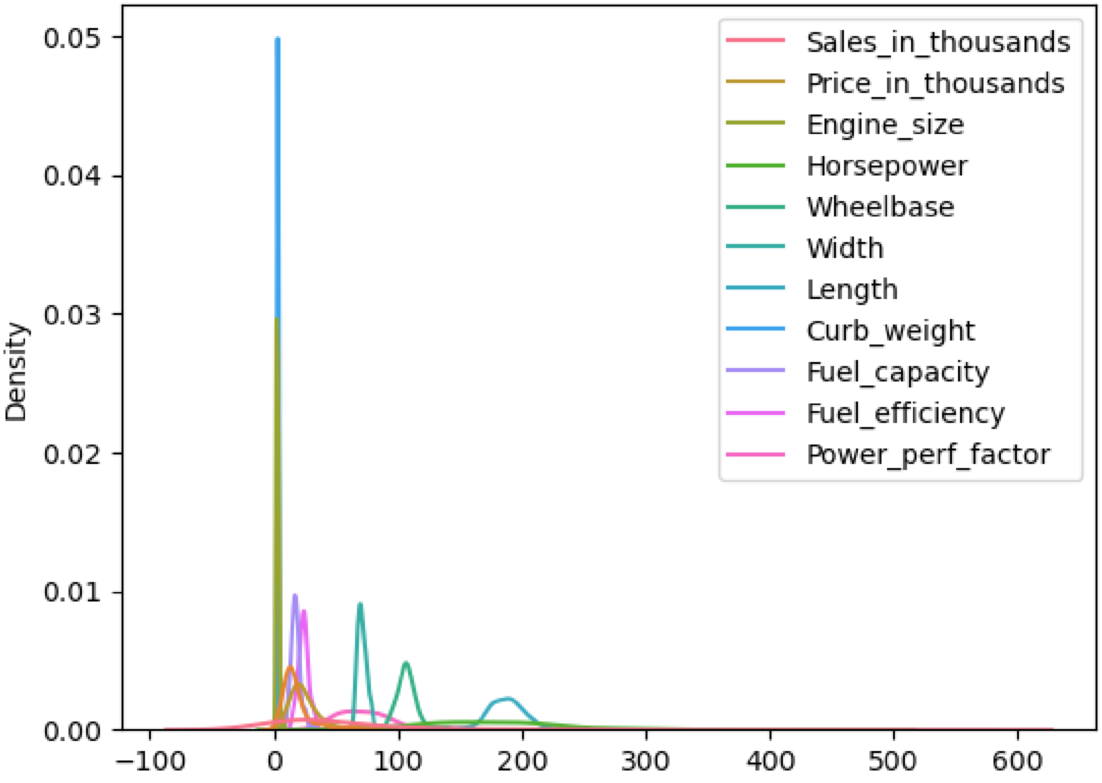
In [13]:

```
plt.figure(figsize=(18,6))
plt.xticks(rotation = 75)
sns.boxplot(data=data)
plt.show()
```



In [14]:

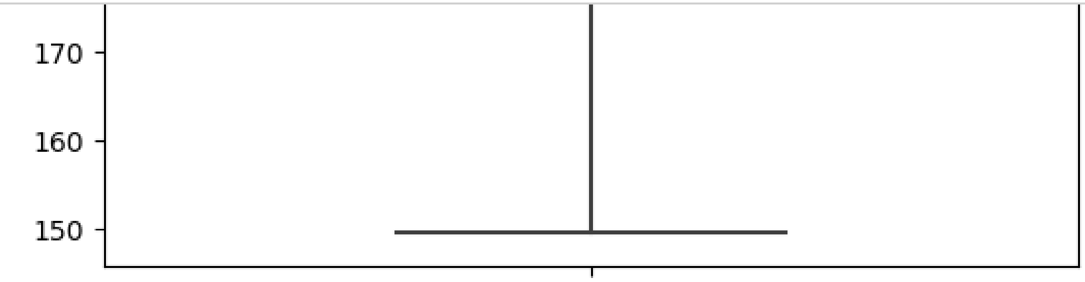
```
sns.kdeplot(data=data)
plt.show()
```

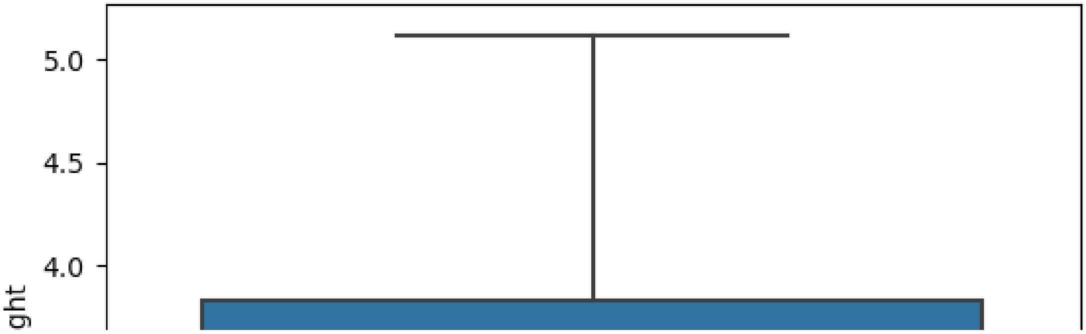


For loop to plot subplots

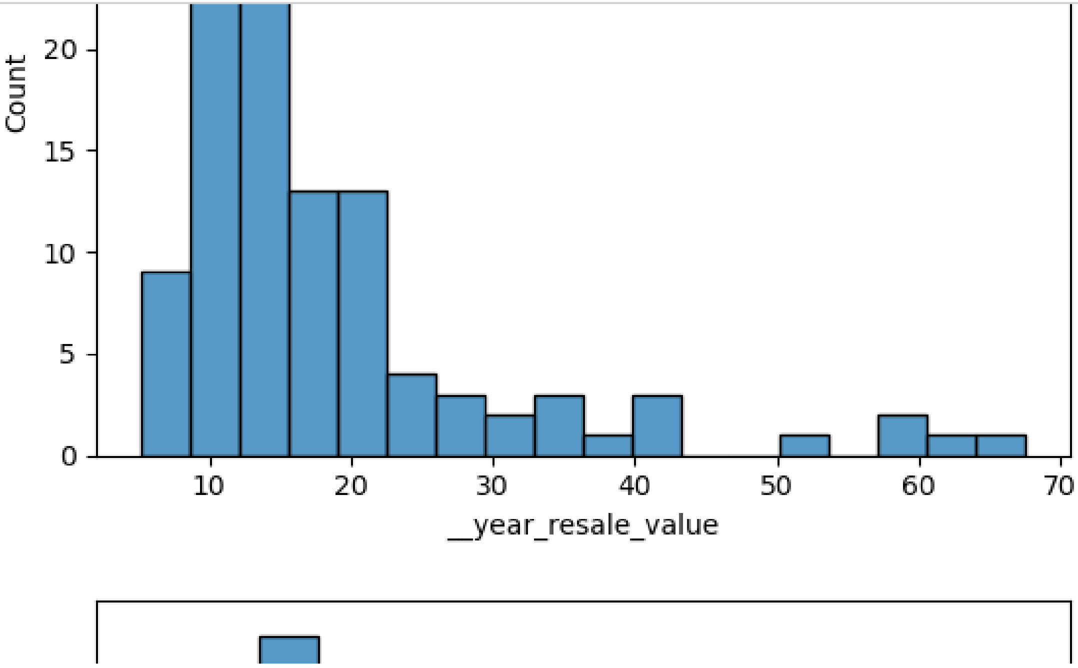
In [15]:

```
for i in data.columns:
    if data[i].dtypes != 'object':
        sns.boxplot(y=data[i])
        plt.show()
```

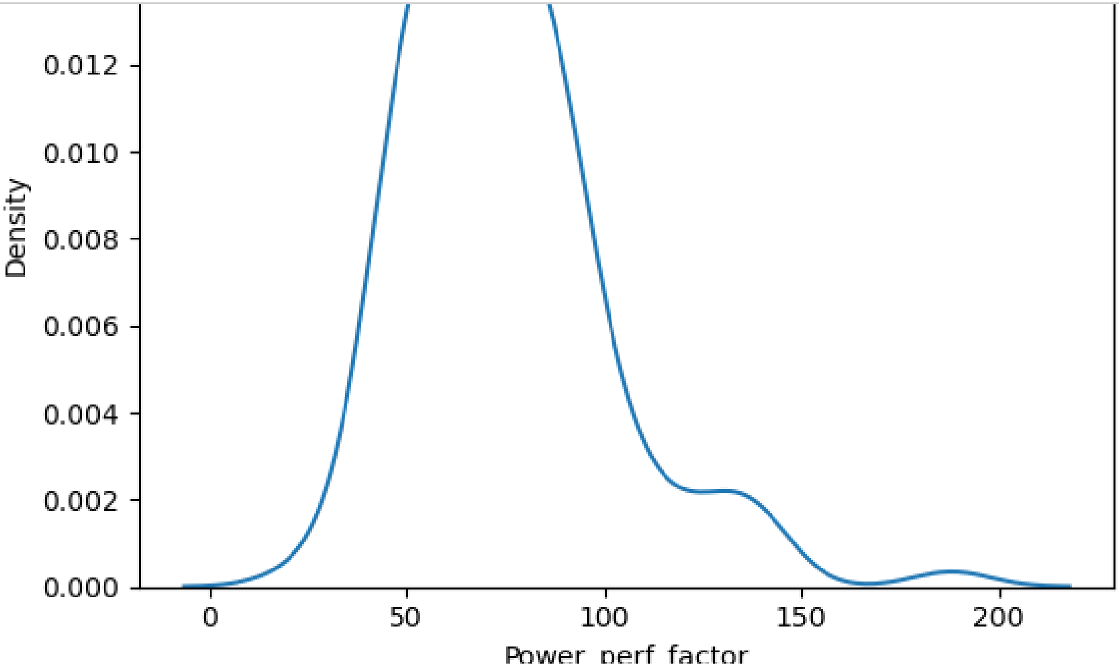




In [16]:
```
for i in data.columns:
    if data[i].dtypes != 'object':
        sns.histplot(x=data[i])
        plt.show()
```



In [17]:
```
for i in data.columns:
    if data[i].dtypes != 'object':
        sns.kdeplot(data=data[i])
        plt.show()
```



Subplots in matplotlib

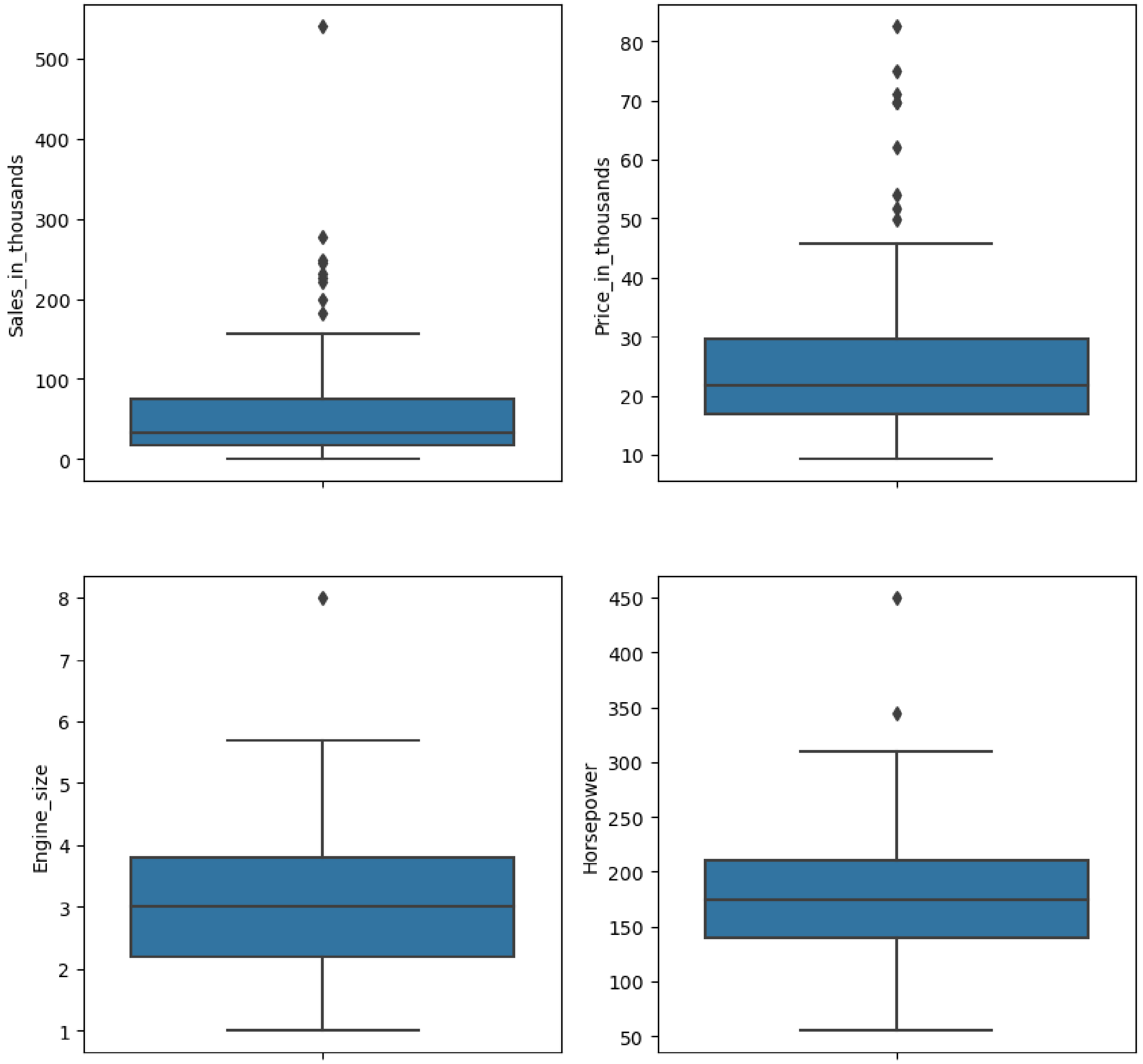
In [18]:
```
plt.figure(figsize=(10,10))
plt.subplot(2,2,1)
sns.boxplot(data=data,y='Sales_in_thousands')

plt.subplot(2,2,2)
sns.boxplot(data=data,y='Price_in_thousands')

plt.subplot(2,2,3)
sns.boxplot(data=data,y='Engine_size')

plt.subplot(2,2,4)
sns.boxplot(data=data,y='Horsepower')

plt.show()
```



Subplots using enumerate

Let's create list in which there is only the columns which we are going to plot

In [19]:
```
features = []
```

In [20]:
```
for i in data.columns:
    if data[i].dtypes != 'object':
        features.append(i)
```

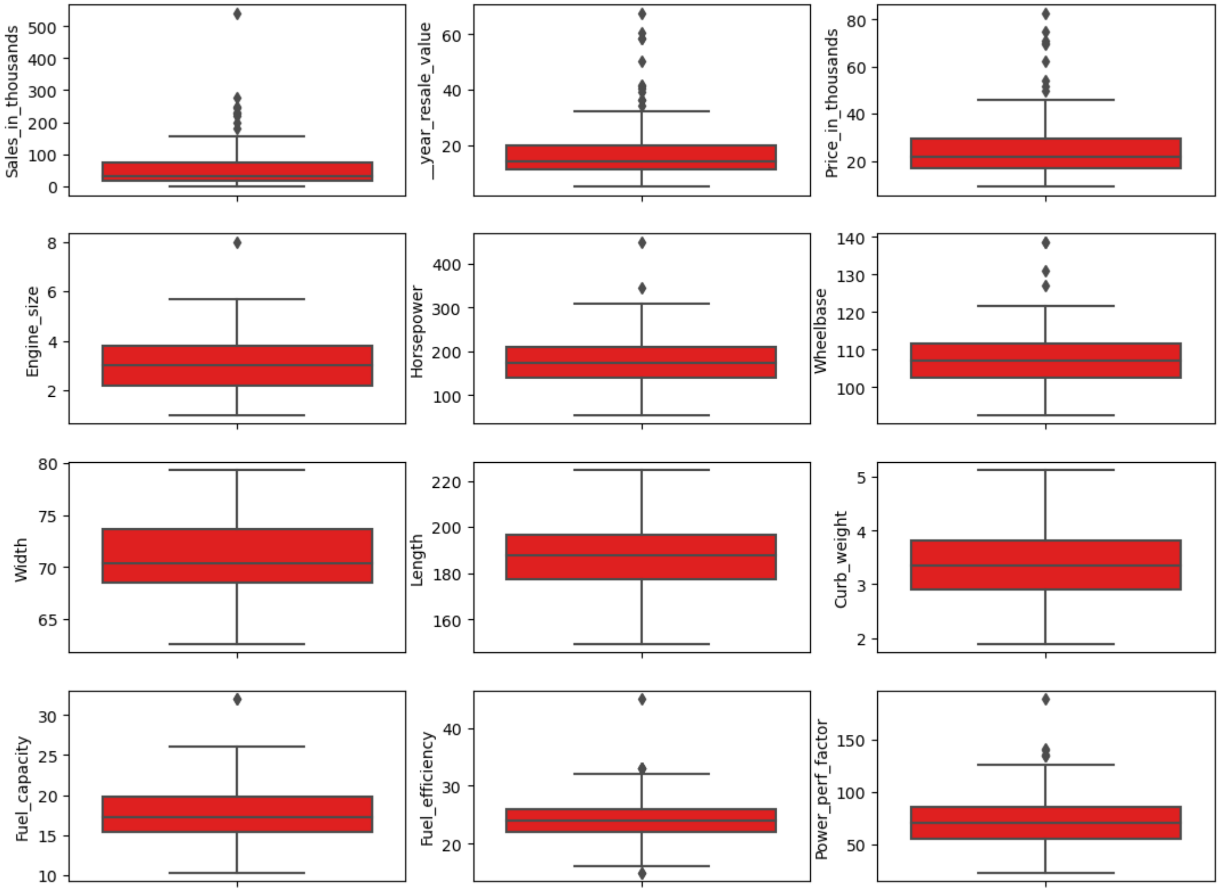
In [21]:
```
features
```

Out[21]:
```
['Sales_in_thousands',
 '__year_resale_value',
 'Price_in_thousands',
 'Engine_size',
 'Horsepower',
 'Wheelbase',
 'Width',
 'Length',
 'Curb_weight',
 'Fuel_capacity',
 'Fuel_efficiency',
 'Power_perf_factor']
```

In [22]:
```
list(enumerate(features))
```

Out[22]:
```
[(0, 'Sales_in_thousands'),
 (1, '__year_resale_value'),
 (2, 'Price_in_thousands'),
 (3, 'Engine_size'),
 (4, 'Horsepower'),
 (5, 'Wheelbase'),
 (6, 'Width'),
 (7, 'Length'),
 (8, 'Curb_weight'),
 (9, 'Fuel_capacity'),
 (10, 'Fuel_efficiency'),
 (11, 'Power_perf_factor')]
```

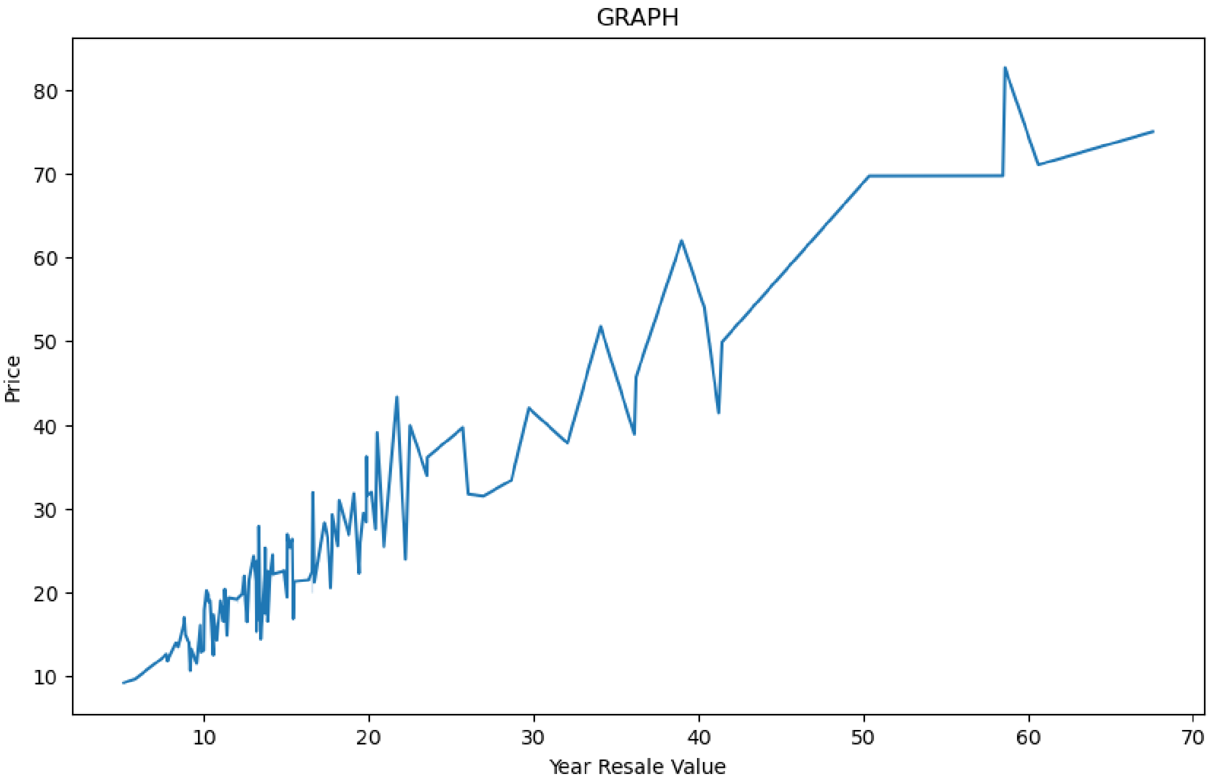
In [23]:
```
plt.figure(figsize=(13,10))
for i in enumerate(features):
    plt.subplot(4,3,i[0]+1)
    sns.boxplot(data=data, y=i[1], color='red')
```



Line charts

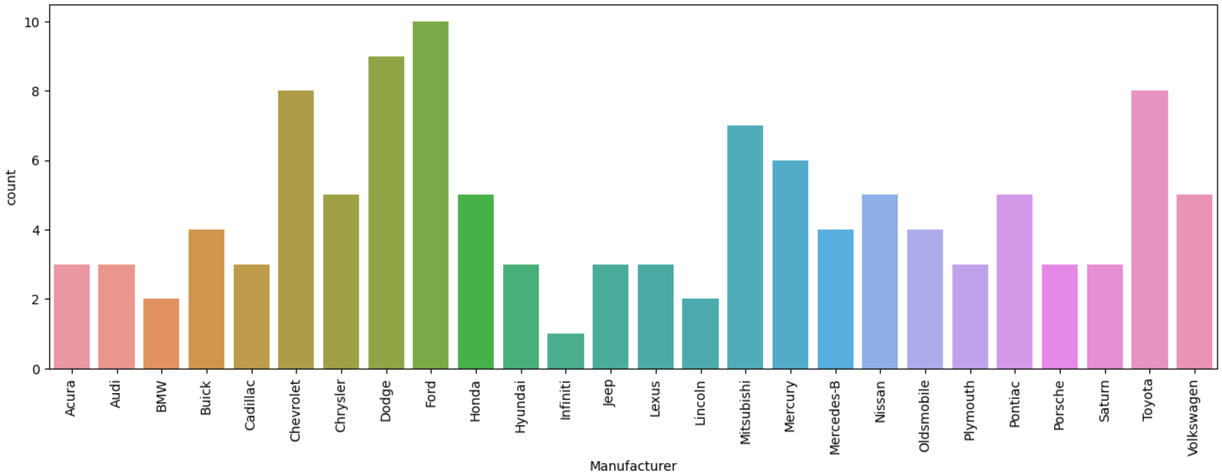
In [24]:

```
plt.figure(figsize=(10,6))
plt.xlabel('Year Resale Value')
plt.ylabel('Price')
plt.title('GRAPH')
sns.lineplot(data=data,x='__year_resale_value', y='Price_in_thousands')
plt.show()
```



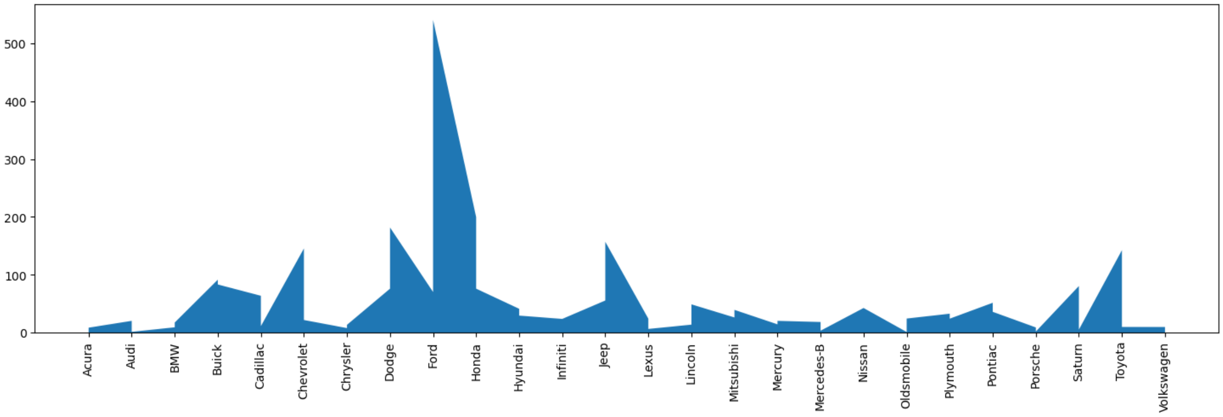
Count plot

In [25]:
```
plt.figure(figsize=(16,5))
plt.xticks(rotation = 90)
sns.countplot(data=data,x='Manufacturer')
plt.show()
```



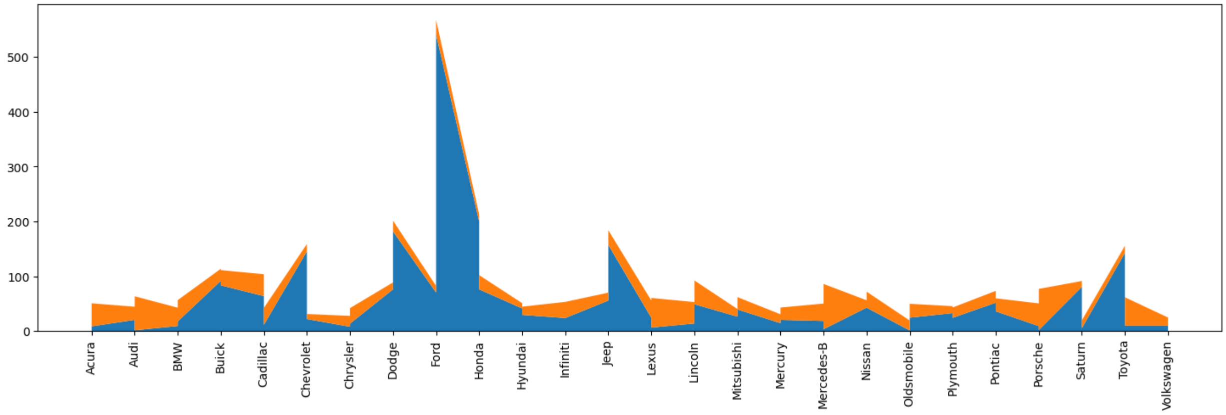
Area chart

In [26]:
```
plt.figure(figsize=(18,5))
plt.xticks(rotation = 90)
plt.stackplot(data.Manufacturer,data.Sales_in_thousands)
plt.show()
```



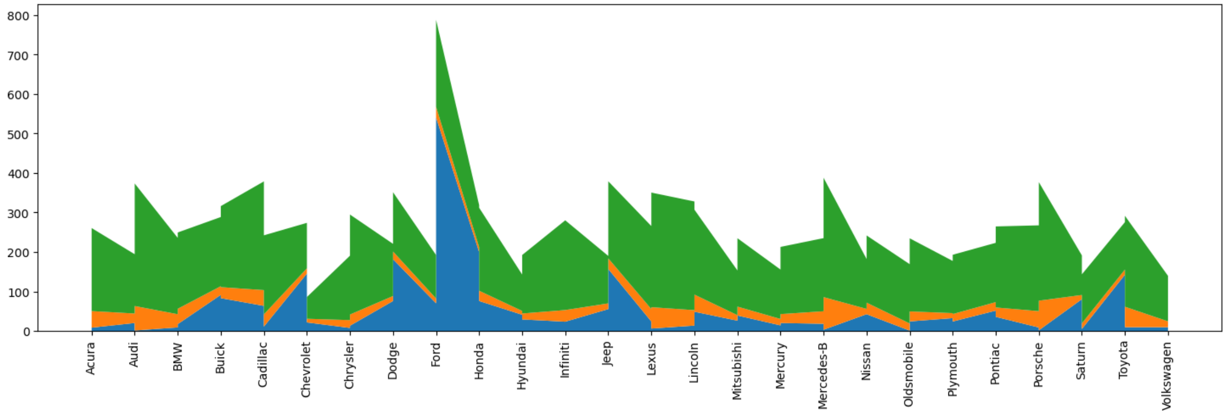
In [27]:

```
plt.figure(figsize=(18,5))
plt.xticks(rotation = 90)
plt.stackplot(data.Manufacturer,data.Sales_in_thousands, data.Price_in_thousands)
plt.show()
```



In [28]:

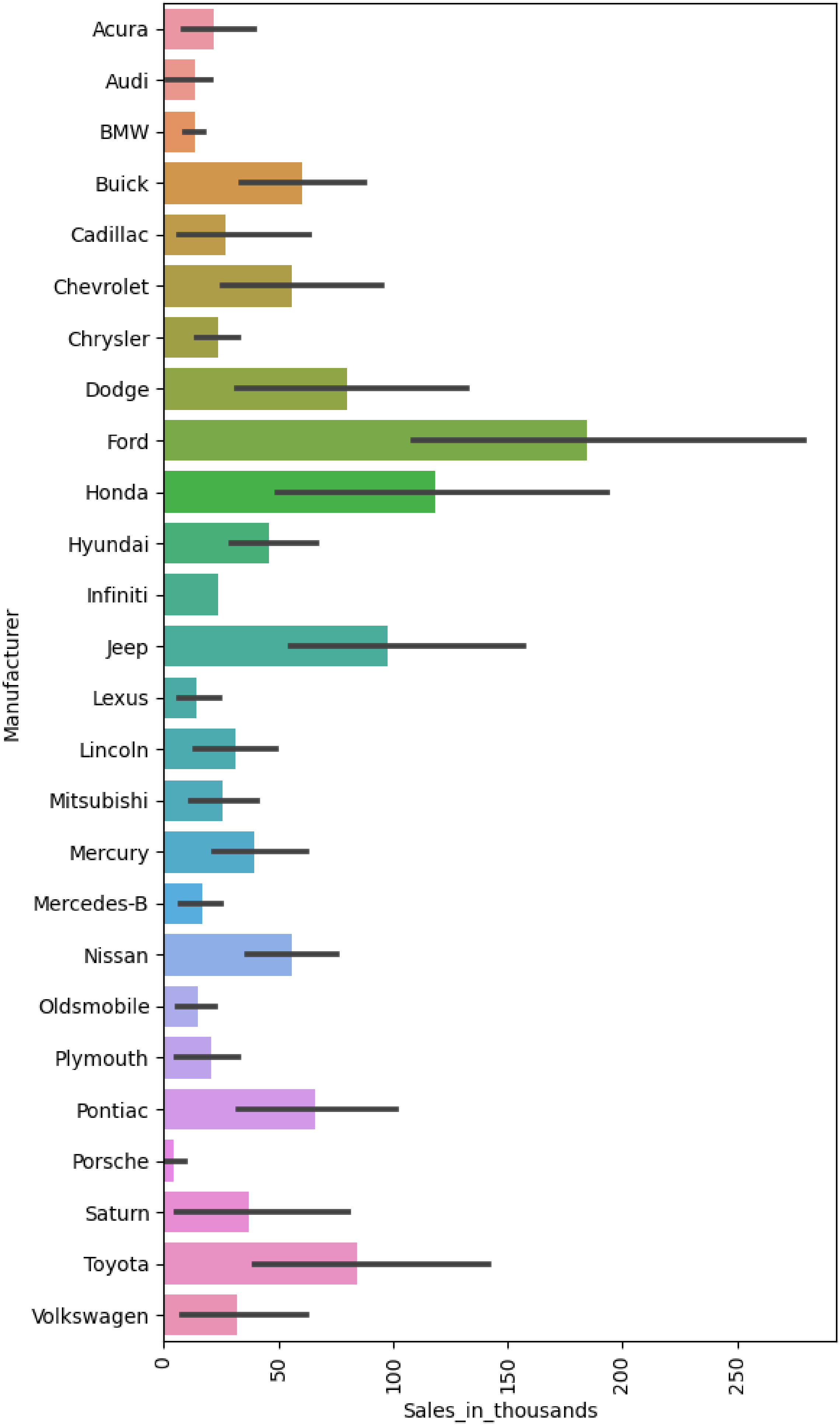
```
plt.figure(figsize=(18,5))
plt.xticks(rotation = 90)
plt.stackplot(data.Manufacturer,data.Sales_in_thousands, data.Price_in_thousands,
plt.show()
```



Horizontal bar chart

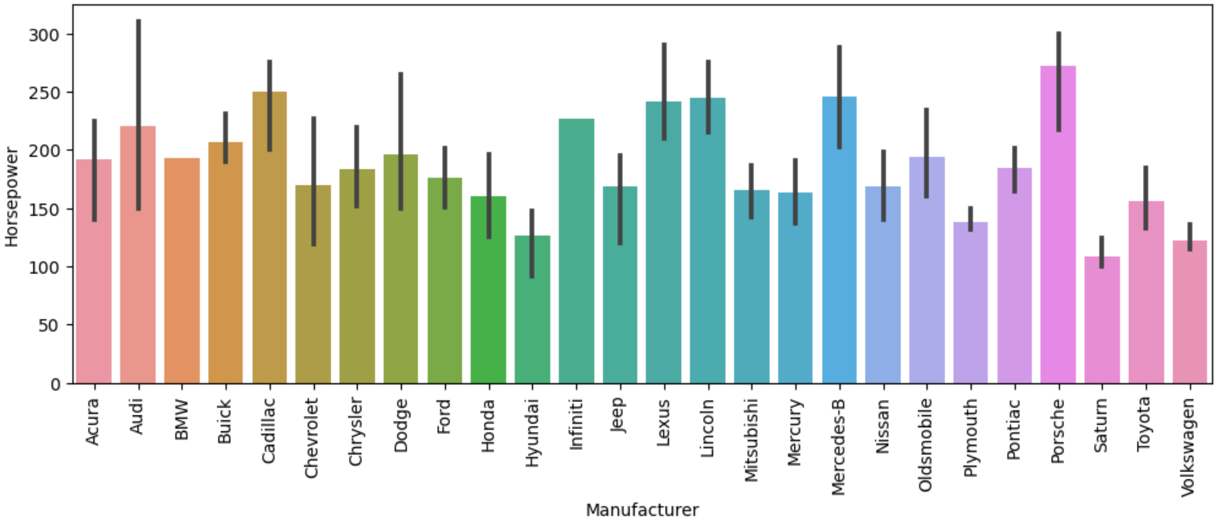
In [29]:

```
plt.figure(figsize=(6,12))
plt.xticks(rotation = 90)
sns.barplot(data=data, x='Sales_in_thousands', y='Manufacturer')
plt.show()
```


Acura
Audi
BMW
Buick
Cadillac
Chevrolet
Chrysler
Dodge
Ford
Honda
Hyundai
Infiniti
Jeep
Lexus
Lincoln
Mitsubishi
Mercury
Mercedes-B
Nissan
Oldsmobile
Plymouth
Pontiac
Porsche
Saturn
Toyota
Volkswagen
Manufacturer
0
50
100
150
200
250
Sales_in_thousands

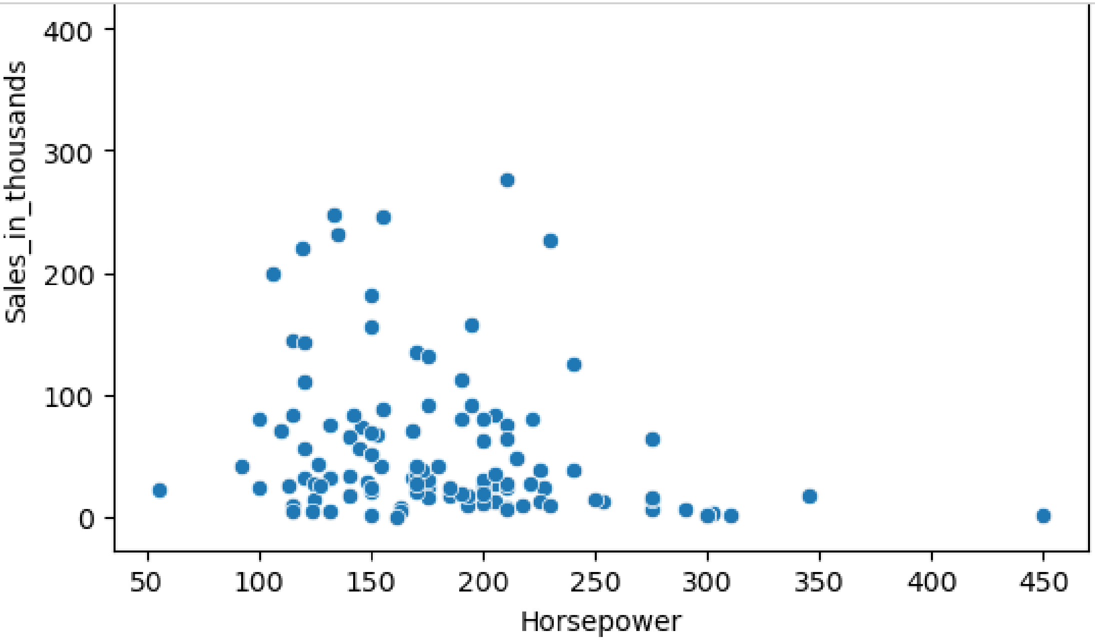
Vertical Bar plot

In [30]:
```
plt.figure(figsize=(12,4))
plt.xticks(rotation = 90)
sns.barplot(data=data, x='Manufacturer', y='Horsepower')
plt.show()
```

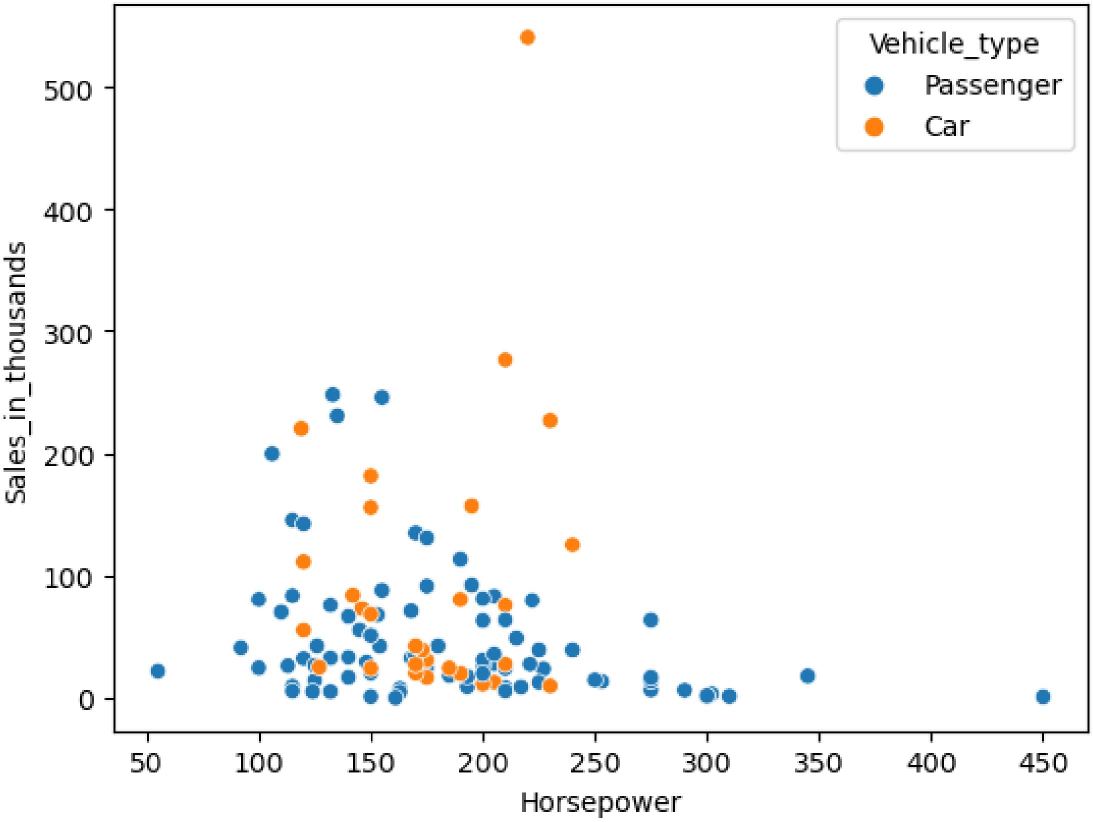


Scatter plot

In [31]:
```
sns.scatterplot(data=data, x='Horsepower', y='Sales_in_thousands')
plt.show()
```



```
In [32]: sns.scatterplot(data=data, x='Horsepower', y='Sales_in_thousands', hue='Vehicle_ty
         plt.show()
```

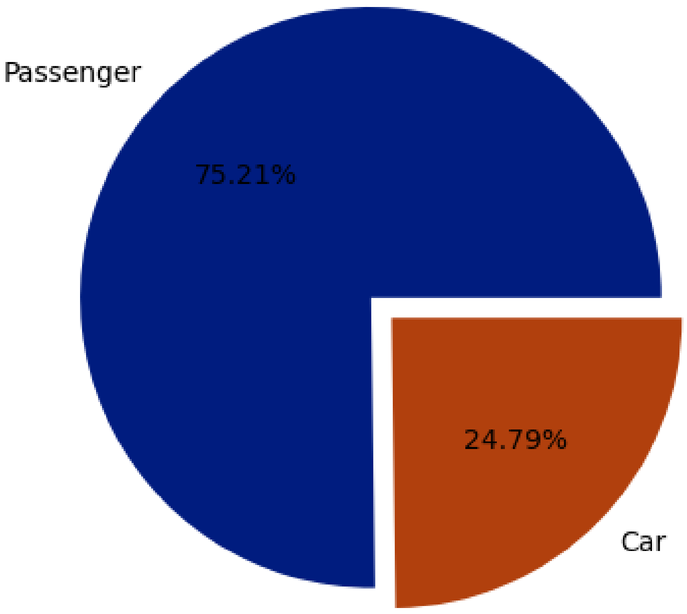


Pie chart

```
In [33]: data.Vehicle_type.value_counts().index
```

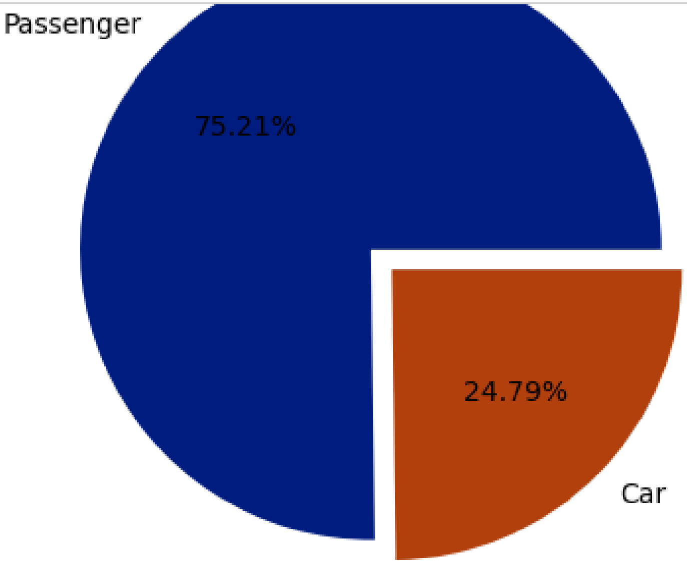
```
Out[33]: Index(['Passenger', 'Car'], dtype='object')
```

```
In [34]: palette_color = sns.color_palette('dark')
In [35]: plt.pie(data=data, x=data.Vehicle_type.value_counts(),explode = [0.1, 0], colors=pa
```



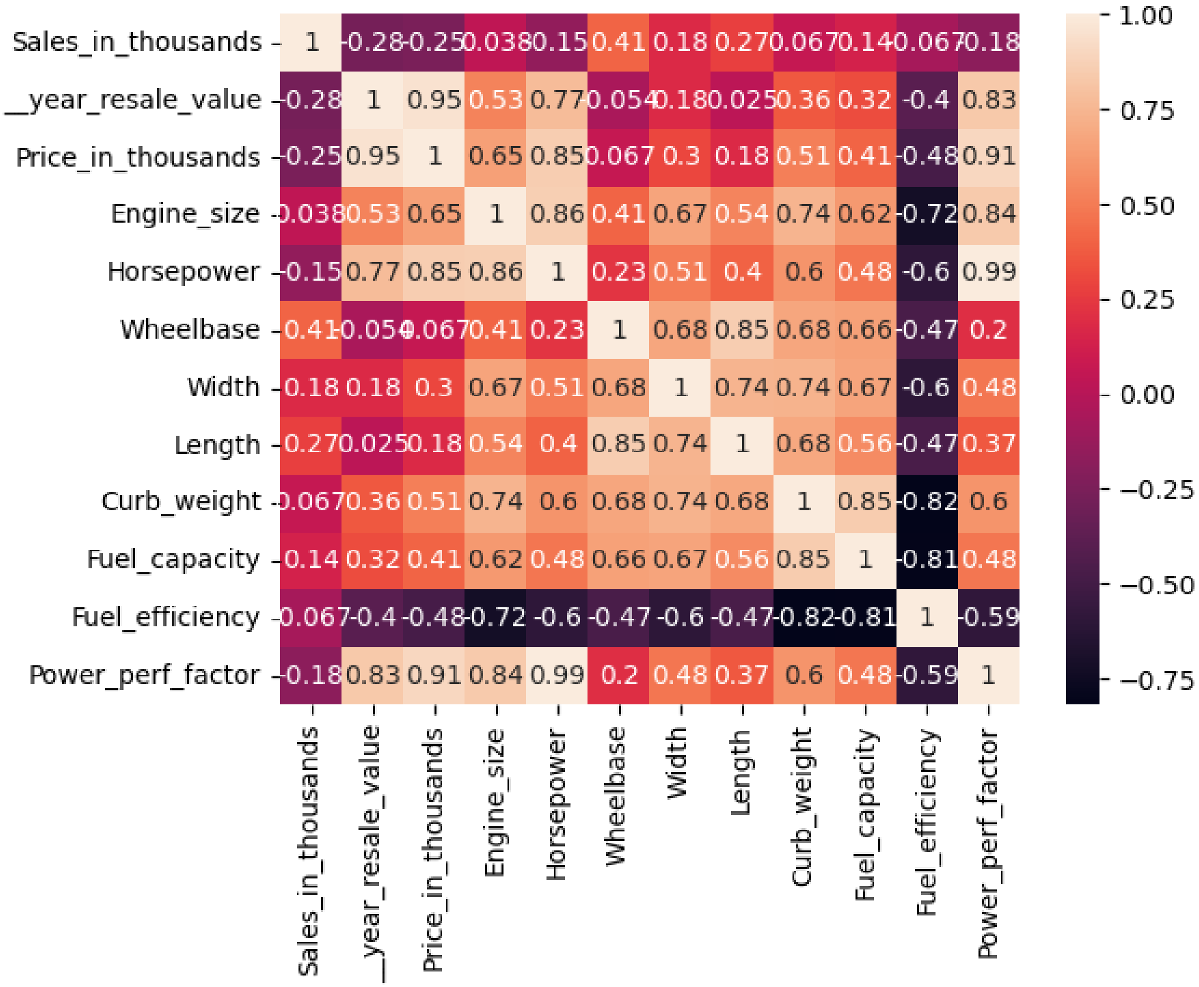
Use ' \ ' for long codes to write in lines seperate

```
In [36]: plt.pie(data=data,\
                 x=data.Vehicle_type.value_counts(),\
                 explode = [0.1, 0], \
                 colors=palette_color,\
                 labels=['Passenger', 'Car'],\
                 autopct='%0.2f%%');
```



Correlation matrix or Heatmap

In [37]:
```
sns.heatmap(data.corr(), annot=True)
plt.show()
```



More cooler approach

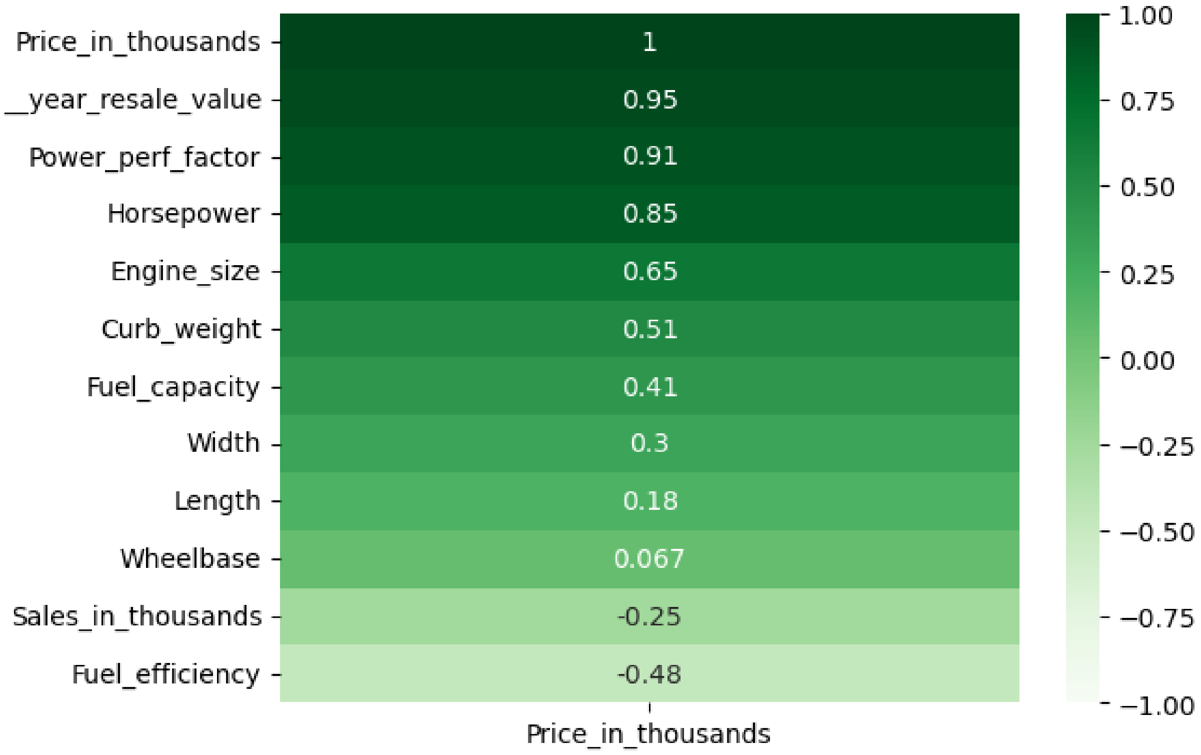
In [38]:
```
data
```

Out[38]:

	Manufacturer	Model	Sales_in_thousands	__year_resale_value	Vehicle_type	Price_in_thousands
0	Acura	Integra	16.919	16.360	Passenger	21.50
1	Acura	TL	39.384	19.875	Passenger	28.40
3	Acura	RL	8.588	29.725	Passenger	42.00
4	Audi	A4	20.397	22.255	Passenger	23.99
5	Audi	A6	18.780	23.555	Passenger	33.95
...	...	...	...	...	...	...
145	Volkswagen	Golf	9.761	11.425	Passenger	14.90
146	Volkswagen	Jetta	83.721	13.240	Passenger	16.70
147	Volkswagen	Passat	51.102	16.725	Passenger	21.20
148	Volkswagen	Cabrio	9.569	16.575	Passenger	19.99
149	Volkswagen	GTI	5.596	13.760	Passenger	17.50

117 rows × 16 columns

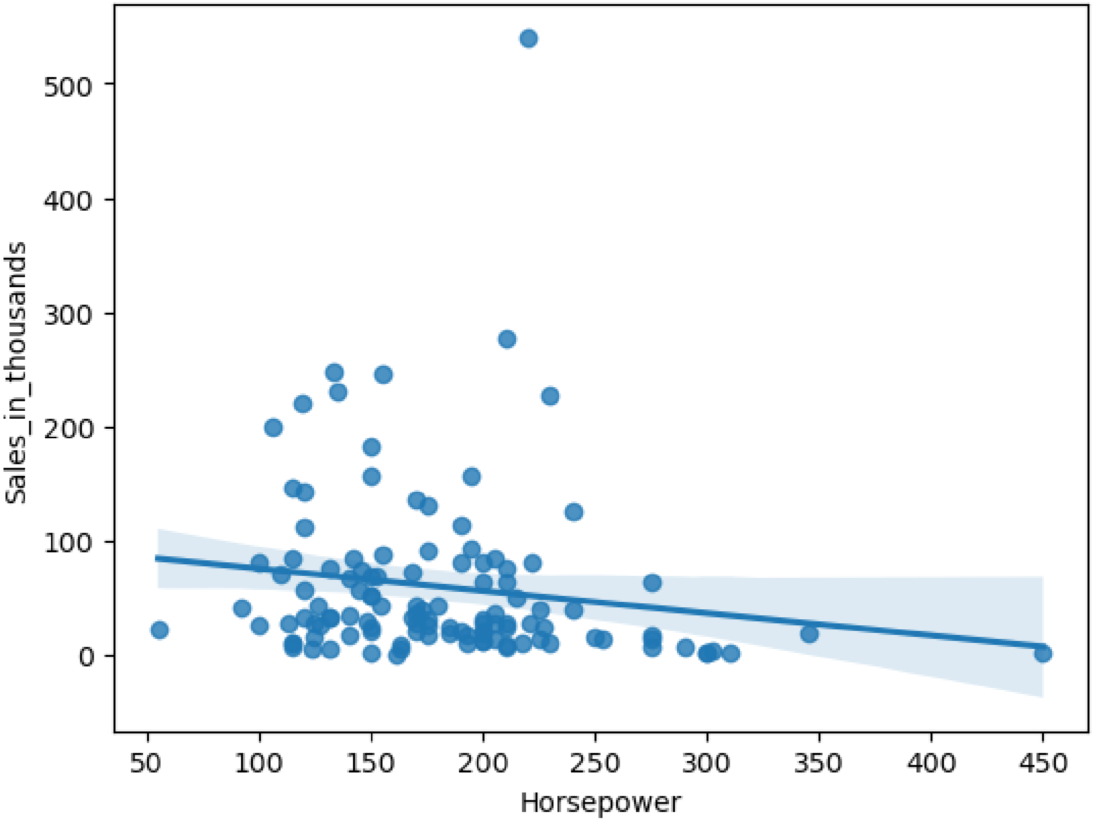
In [39]:
```
sns.heatmap(data.corr()[["Price_in_thousands"]].sort_values(by="Price_in_thousands'
plt.plot();
```



Regplot

In [40]:

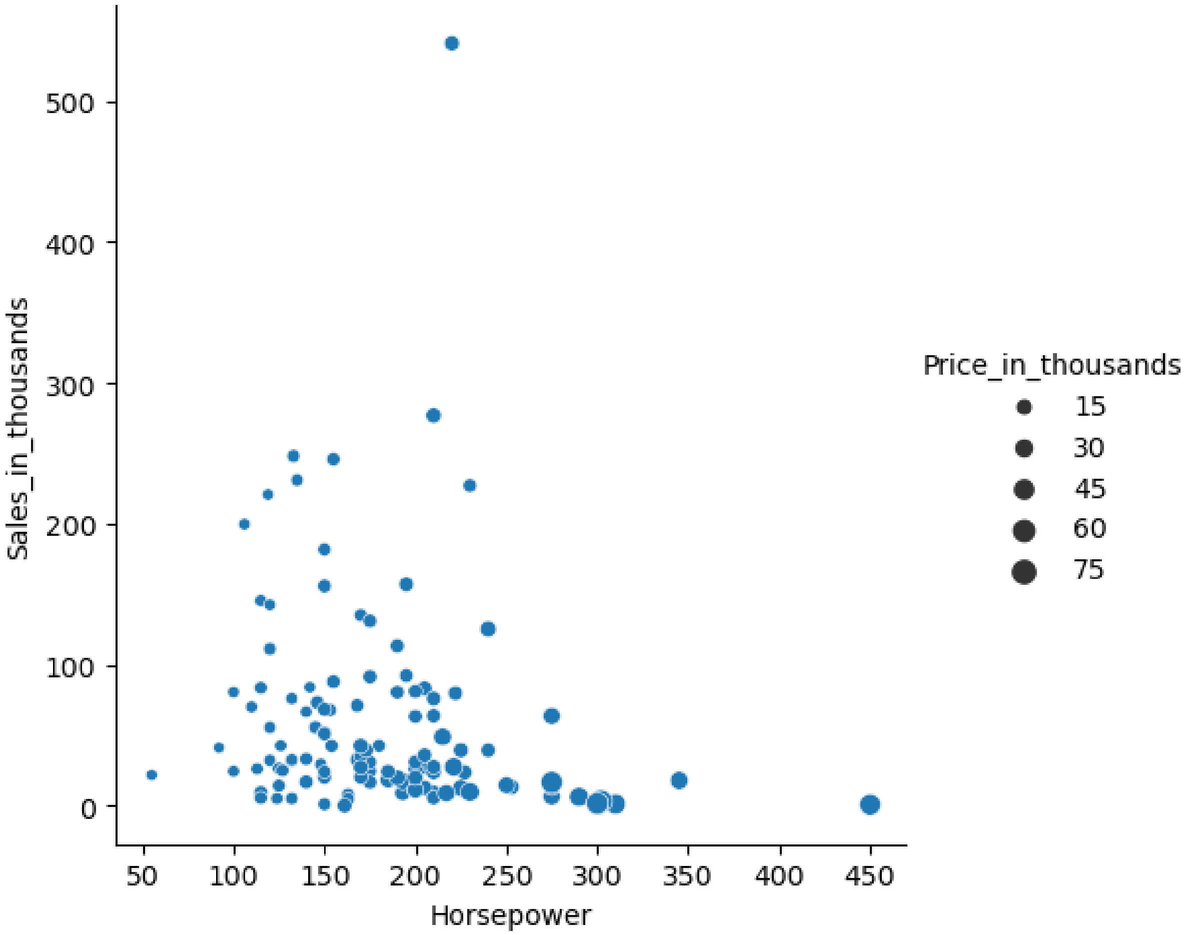
```
sns.regplot(data = data, x = 'Horsepower', y= 'Sales_in_thousands')
plt.show()
```



Relplot

In [41]:

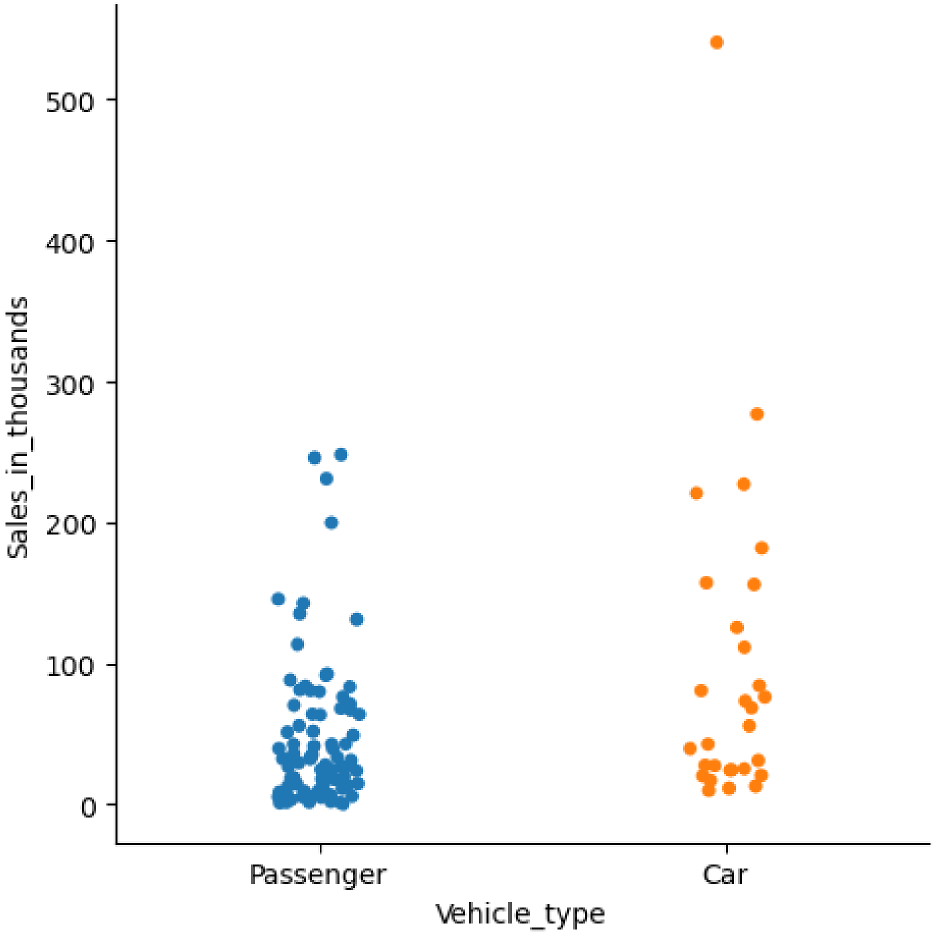
```
sns.relplot(data = data, x = 'Horsepower', y= 'Sales_in_thousands', size='Price_in_
plt.show()
```



Catplot

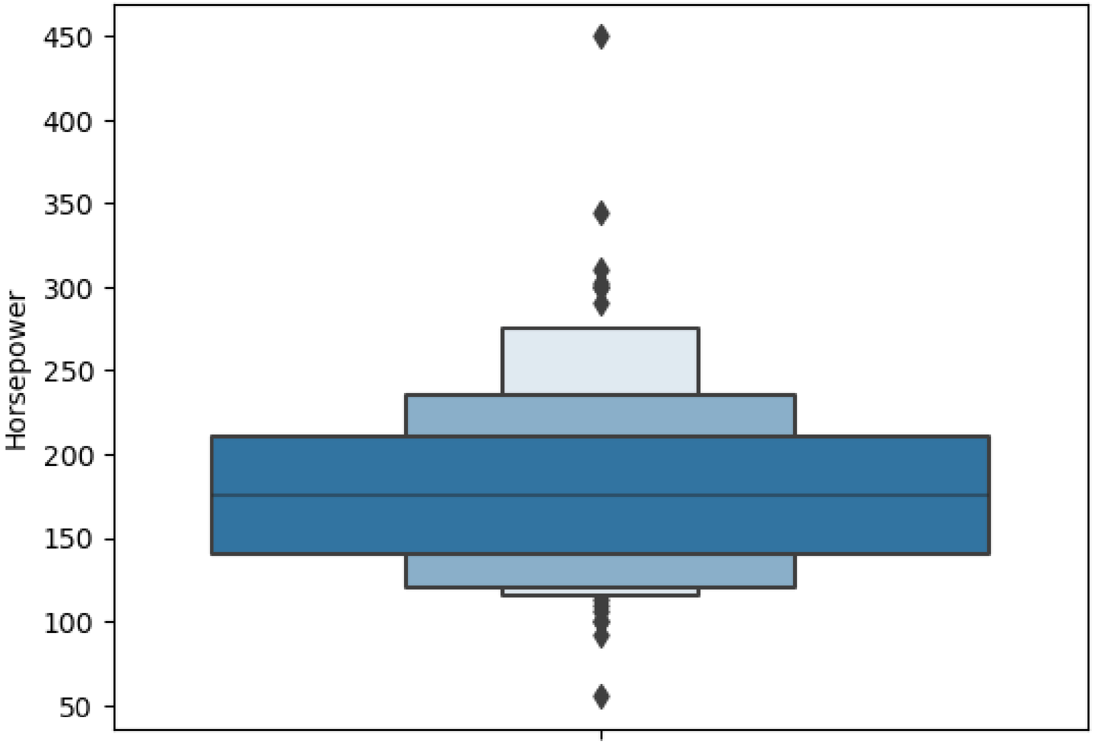
In [42]:

```
sns.catplot(data = data, x = 'Vehicle_type', y= 'Sales_in_thousands')
plt.show()
```



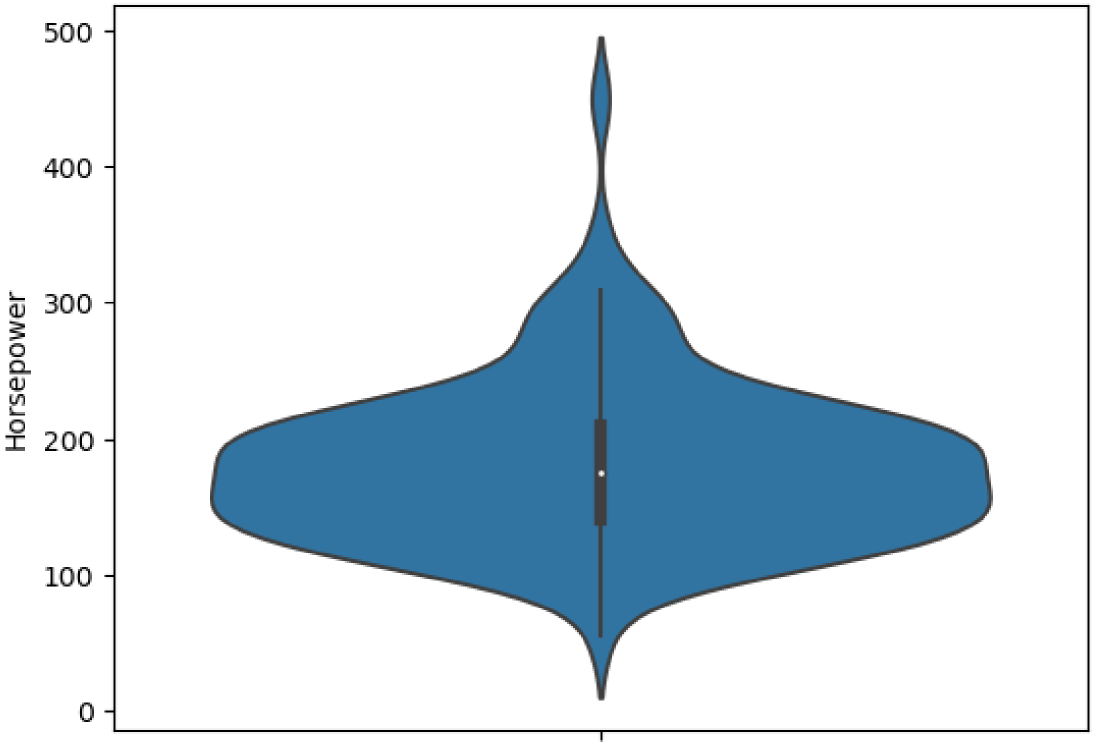
Boxen plot

In [43]:
```
sns.boxenplot(data=data, y='Horsepower')
plt.show()
```



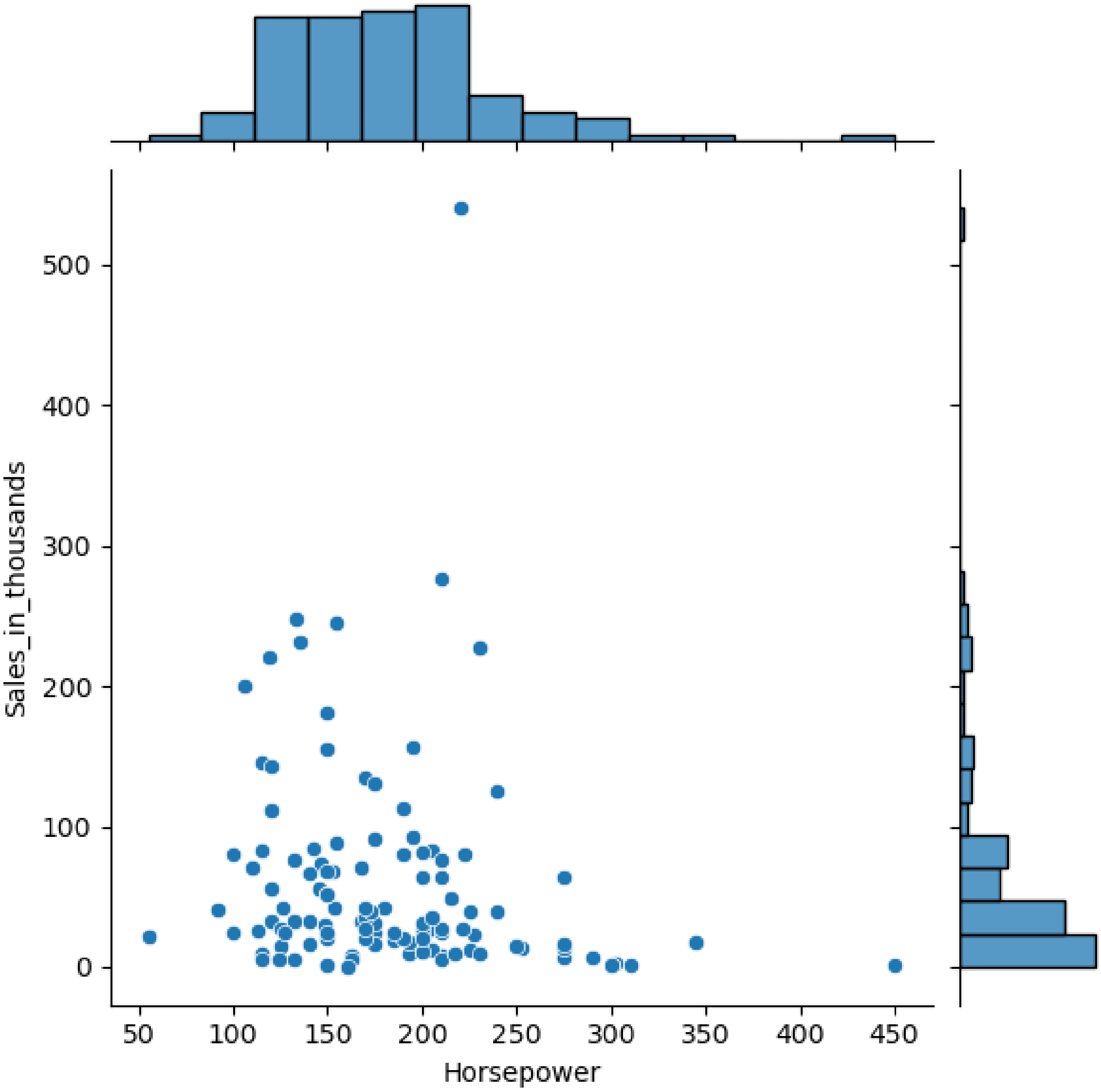
violin plot

```
In [44]: sns.violinplot(data=data, y='Horsepower')
         plt.show()
```

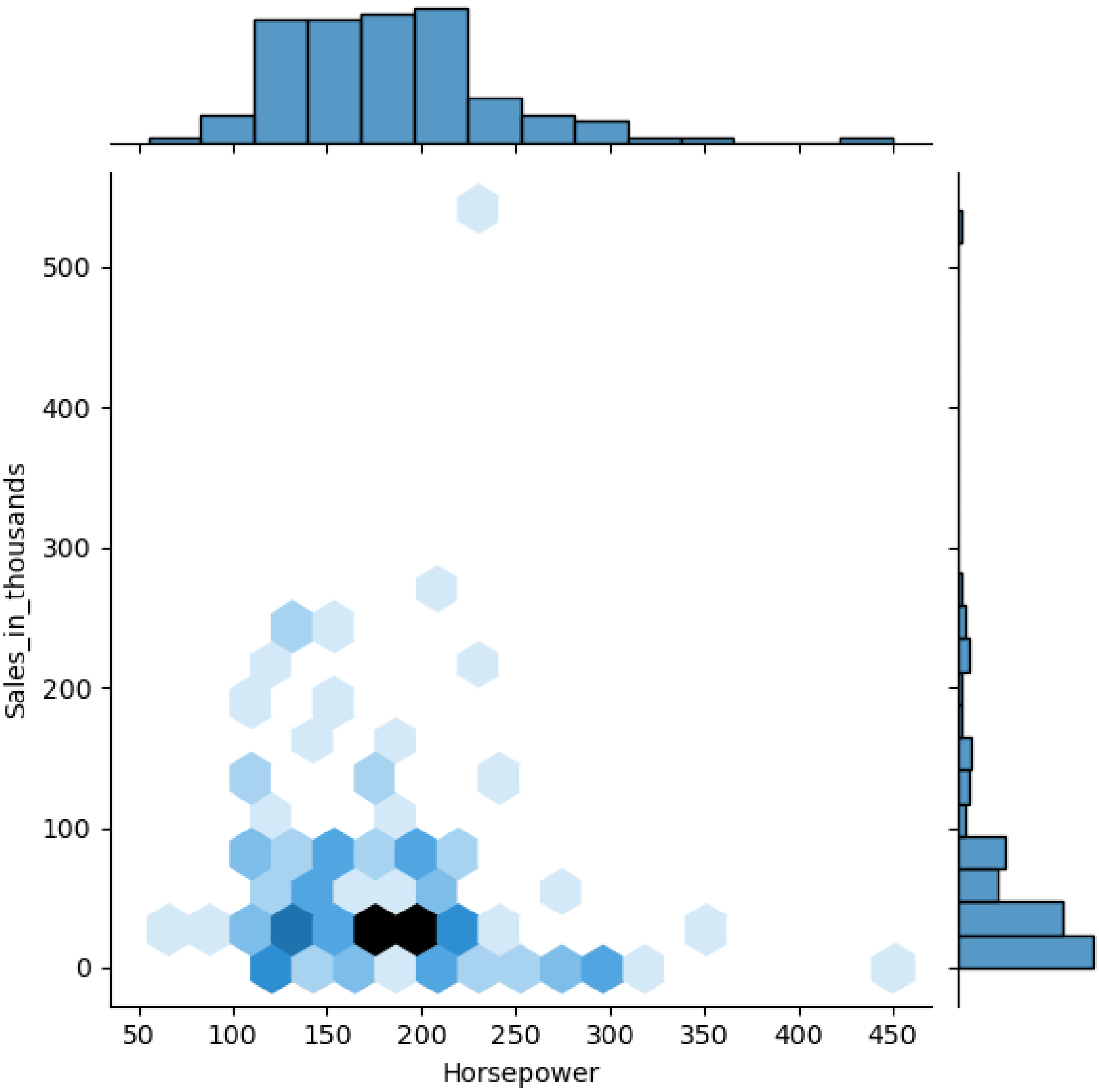


Joint plot

```
In [45]: sns.jointplot(data = data, x = 'Horsepower', y= 'Sales_in_thousands')
         plt.show()
```



In [46]:
```
sns.jointplot(data = data, x = 'Horsepower', y= 'Sales_in_thousands', kind='hex')
plt.show()
```

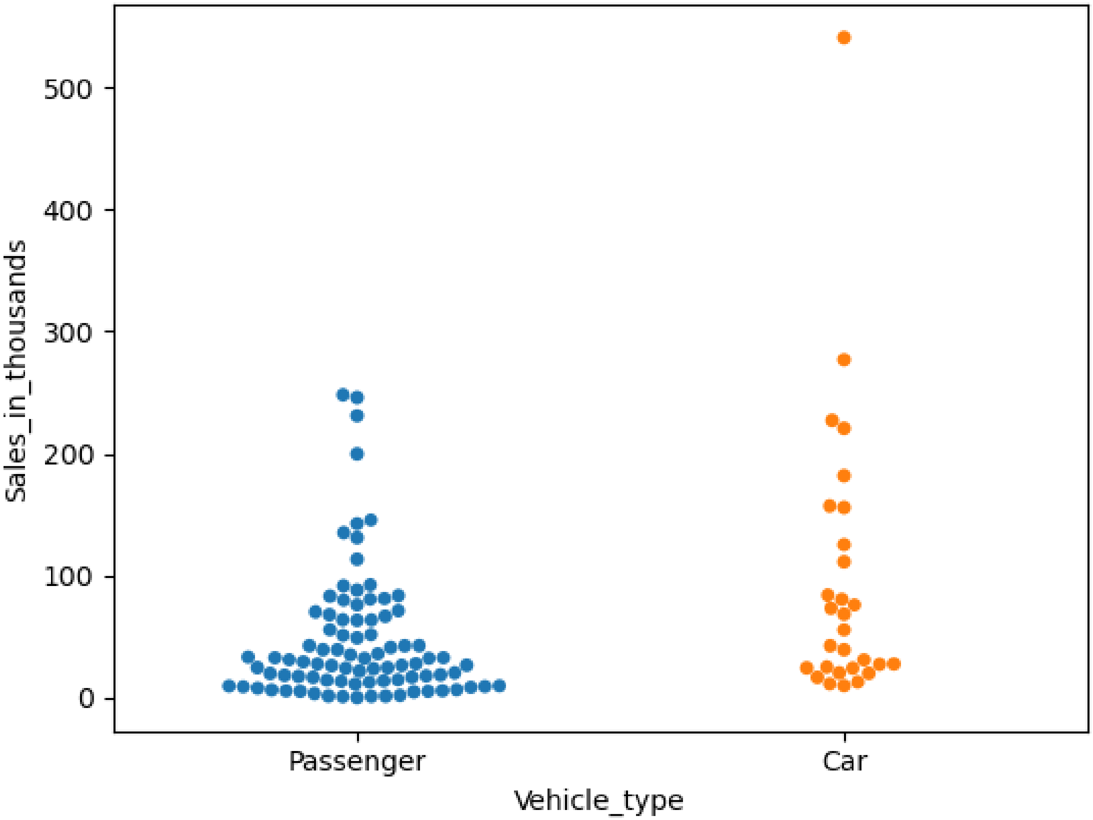


Swarm plot

In [47]:
```
data.columns
```

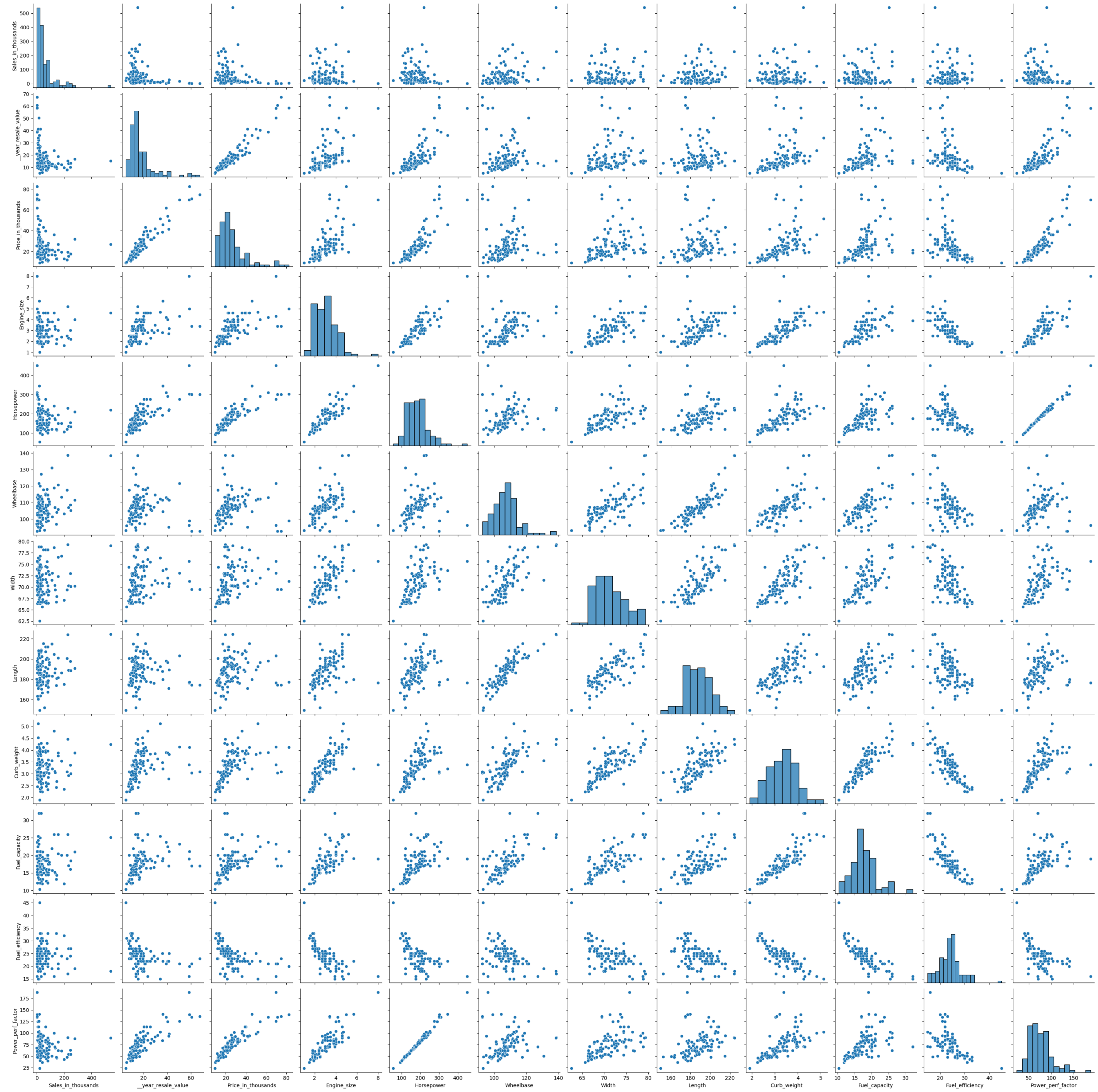
Out[47]:
```
Index(['Manufacturer', 'Model', 'Sales_in_thousands', '__year_resale_value',
       'Vehicle_type', 'Price_in_thousands', 'Engine_size', 'Horsepower',
       'Wheelbase', 'Width', 'Length', 'Curb_weight', 'Fuel_capacity',
       'Fuel_efficiency', 'Latest_Launch', 'Power_perf_factor'],
      dtype='object')
```

In [48]:
```
sns.swarmplot(data = data, x = 'Vehicle_type', y= 'Sales_in_thousands')
plt.show()
```



Pair Plot

In [49]:
```
sns.pairplot(data=data)
plt.show()
```



```
In [50]: data
```

```
Out[50]:
     Manufacturer  Model    Sales_in_thousands  __year_resale_value  Vehicle_type  Price_in_thousands
  0         Acura  Integra              16.919               16.360     Passenger               21.50
  1         Acura       TL              39.384               19.875     Passenger               28.40
  3         Acura       RL               8.588               29.725     Passenger               42.00
  4          Audi       A4              20.397               22.255     Passenger               23.99
  5          Audi       A6              18.780               23.555     Passenger               33.95
...           ...      ...                 ...                  ...           ...                 ...
145    Volkswagen     Golf               9.761               11.425     Passenger               14.90
146    Volkswagen    Jetta              83.721               13.240     Passenger               16.70
147    Volkswagen   Passat              51.102               16.725     Passenger               21.20
148    Volkswagen   Cabrio               9.569               16.575     Passenger               19.99
149    Volkswagen      GTI               5.596               13.760     Passenger               17.50

117 rows × 16 columns
```