Project as Exploratory Data Analysis (EDA) by HARDEEP SINGH

Overview

In [1]:
```
#  Understanding how EDA is done in 'Python' using the 'Jupyter'
#  Various baisc and simple steps involved in the Exploratory Data Analysis (EDA)
#  Performing Exploratory Data Analysis (EDA) on a given dataset related to Car_Sales
```

Contents

In [2]:
```
# 1. Importing the essential Python Libraries

# 2. Loading the Dataset related to Car_Sales

# 3. Pattern of the given Data

# 4. Handling of Duplicates

# 5. Handling of Outliers

# 6. Handle Missing Values

# 7. Univariate Analysis

# 8. Bivariate Analysis

# 9. Multivariate Analysis
```

1. Importing the essential Python Libraries

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

2. Loading the Dataset related to Car_Sales

In [4]:
```
# Loading the Dataset related to Car_Sales using Pandas
data = pd.read_csv('car_sales.csv')
```

In [5]:
```
# Calling the above created variable of data
data
```

Out[5]:

	Manufacturer	Model	Sales_in_thousands	__year_resale_value	Vehicle_type	Price_in_thousands	Engine_size	Horsepower	Wheelbase	Wid
0	Acura	Integra	16.919	16.360	Passenger	21.50	1.8	140.0	101.2	67
1	Acura	TL	39.384	19.875	Passenger	28.40	3.2	225.0	108.1	70
2	Acura	CL	14.114	18.225	Passenger	NaN	3.2	225.0	106.9	70
3	Acura	RL	8.588	29.725	Passenger	42.00	3.5	210.0	114.6	71
4	Audi	A4	20.397	22.255	Passenger	23.99	1.8	150.0	102.6	68
...	...	...	...	...	...	...	...	...	...	
152	Volvo	V40	3.545	NaN	Passenger	24.40	1.9	160.0	100.5	67
153	Volvo	S70	15.245	NaN	Passenger	27.50	2.4	168.0	104.9	69
154	Volvo	V70	17.531	NaN	Passenger	28.80	2.4	168.0	104.9	69
155	Volvo	C70	3.493	NaN	Passenger	45.50	2.3	236.0	104.9	71
156	Volvo	S80	18.969	NaN	Passenger	36.00	2.9	201.0	109.9	72

157 rows × 16 columns

3. Pattern of the given Data

In [6]:
```
# Displaying First Ten Data Observations
data.head(10)
```

Out[6]:

	Manufacturer	Model	Sales_in_thousands	__year_resale_value	Vehicle_type	Price_in_thousands	Engine_size	Horsepower	Wheelbase	Width
0	Acura	Integra	16.919	16.360	Passenger	21.50	1.8	140.0	101.2	67.3
1	Acura	TL	39.384	19.875	Passenger	28.40	3.2	225.0	108.1	70.3
2	Acura	CL	14.114	18.225	Passenger	NaN	3.2	225.0	106.9	70.6
3	Acura	RL	8.588	29.725	Passenger	42.00	3.5	210.0	114.6	71.4
4	Audi	A4	20.397	22.255	Passenger	23.99	1.8	150.0	102.6	68.2
5	Audi	A6	18.780	23.555	Passenger	33.95	2.8	200.0	108.7	76.1
6	Audi	A8	1.380	39.000	Passenger	62.00	4.2	310.0	113.0	74.0
7	BMW	323i	19.747	NaN	Passenger	26.99	2.5	170.0	107.3	68.4
8	BMW	328i	9.231	28.675	Passenger	33.40	2.8	193.0	107.3	68.5
9	BMW	528i	17.527	36.125	Passenger	38.90	2.8	193.0	111.4	70.9

In [7]:
```
# Displaying Last Ten Data Observations
data.tail(10)
```

Out[7]:

	Manufacturer	Model	Sales_in_thousands	__year_resale_value	Vehicle_type	Price_in_thousands	Engine_size	Horsepower	Wheelbase	Wid
147	Volkswagen	Passat	51.102	16.725	Passenger	21.20	1.8	150.0	106.4	68
148	Volkswagen	Cabrio	9.569	16.575	Passenger	19.99	2.0	115.0	97.4	66
149	Volkswagen	GTI	5.596	13.760	Passenger	17.50	2.0	115.0	98.9	68
150	Volkswagen	Beetle	49.463	NaN	Passenger	15.90	2.0	115.0	98.9	67
151	Volvo	S40	16.957	NaN	Passenger	23.40	1.9	160.0	100.5	67
152	Volvo	V40	3.545	NaN	Passenger	24.40	1.9	160.0	100.5	67
153	Volvo	S70	15.245	NaN	Passenger	27.50	2.4	168.0	104.9	69
154	Volvo	V70	17.531	NaN	Passenger	28.80	2.4	168.0	104.9	69
155	Volvo	C70	3.493	NaN	Passenger	45.50	2.3	236.0	104.9	71
156	Volvo	S80	18.969	NaN	Passenger	36.00	2.9	201.0	109.9	72

In [8]:
```
# Displaying the Number of Variables and Observations
data.shape
```

Out[8]: (157, 16)

In [9]:
```
# So in this dataset (data), there are 157 Rows and 16 Columns
```

In [10]:
```
data.dtypes
# It will display the Variables Names and their Data Types
```

Out[10]:
```
Manufacturer            object
Model                   object
Sales_in_thousands     float64
__year_resale_value    float64
Vehicle_type            object
Price_in_thousands     float64
Engine_size            float64
Horsepower             float64
Wheelbase              float64
Width                  float64
Length                 float64
Curb_weight            float64
Fuel_capacity          float64
Fuel_efficiency        float64
Latest_Launch           object
Power_perf_factor      float64
dtype: object
```

In [11]:
```
# Counting the number of Non-Missing Values for each Variable
data.count()
```

Out[11]:
```
Manufacturer          157
Model                 157
Sales_in_thousands    157
__year_resale_value   121
Vehicle_type          157
Price_in_thousands    155
Engine_size           156
Horsepower            156
Wheelbase             156
Width                 156
Length                156
Curb_weight           155
Fuel_capacity         156
Fuel_efficiency       154
Latest_Launch         157
Power_perf_factor     155
dtype: int64
```

In [12]:
```
data.describe()
# It will give statistical information of all numerical features in our dataset
# Count - count of the values
# mean - average of the values
# std - standard deviation
# min - minimum values as compared to the other values
# 25% - first quartile
# 50% - second quartile of near to the median
# 75% - third quartile
# max - maximum values as compared to the other values
```

Out[12]:

	Sales_in_thousands	__year_resale_value	Price_in_thousands	Engine_size	Horsepower	Wheelbase	Width	Length	Curb_weight
count	157.000000	121.000000	155.000000	156.000000	156.000000	156.000000	156.000000	156.000000	155.000000
mean	52.998076	18.072975	27.390755	3.060897	185.948718	107.487179	71.150000	187.343590	3.378026
std	68.029422	11.453384	14.351653	1.044653	56.700321	7.641303	3.451872	13.431754	0.630502
min	0.110000	5.160000	9.235000	1.000000	55.000000	92.600000	62.600000	149.400000	1.895000
25%	14.114000	11.260000	18.017500	2.300000	149.500000	103.000000	68.400000	177.575000	2.971000
50%	29.450000	14.180000	22.799000	3.000000	177.500000	107.000000	70.550000	187.900000	3.342000
75%	67.956000	19.875000	31.947500	3.575000	215.000000	112.200000	73.425000	196.125000	3.799500
max	540.561000	67.550000	85.500000	8.000000	450.000000	138.700000	79.900000	224.500000	5.572000

In [13]:
```
# We can also check this distribution for a particular numeric column
data.Sales_in_thousands.describe()
```

Out[13]:
```
count    157.000000
mean      52.998076
std       68.029422
min        0.110000
25%       14.114000
50%       29.450000
75%       67.956000
max      540.561000
Name: Sales_in_thousands, dtype: float64
```

In [14]:
```
data.describe(include='all')
# In this way, we can check for the all the attributes of the Dataset
```

Out[14]:

	Manufacturer	Model	Sales_in_thousands	__year_resale_value	Vehicle_type	Price_in_thousands	Engine_size	Horsepower	Wheelbase
count	157	157	157.000000	121.000000	157	155.000000	156.000000	156.000000	156.000000
unique	30	156	NaN	NaN	2	NaN	NaN	NaN	NaN
top	Dodge	Neon	NaN	NaN	Passenger	NaN	NaN	NaN	NaN
freq	11	2	NaN	NaN	116	NaN	NaN	NaN	NaN
mean	NaN	NaN	52.998076	18.072975	NaN	27.390755	3.060897	185.948718	107.487179
std	NaN	NaN	68.029422	11.453384	NaN	14.351653	1.044653	56.700321	7.641303
min	NaN	NaN	0.110000	5.160000	NaN	9.235000	1.000000	55.000000	92.600000
25%	NaN	NaN	14.114000	11.260000	NaN	18.017500	2.300000	149.500000	103.000000
50%	NaN	NaN	29.450000	14.180000	NaN	22.799000	3.000000	177.500000	107.000000
75%	NaN	NaN	67.956000	19.875000	NaN	31.947500	3.575000	215.000000	112.200000
max	NaN	NaN	540.561000	67.550000	NaN	85.500000	8.000000	450.000000	138.700000

In [15]:
```
# For checking the complete summary of the Dataset which will include data types,shape and memory storage, attrib
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
```

4. Handling of Duplicates

In [16]:
```
data.duplicated()
# It will show the list of duplicated values
```

Out[16]:
```
0      False
1      False
2      False
3      False
4      False
       ...
152    False
153    False
154    False
155    False
156    False
Length: 157, dtype: bool
```

In [17]:
```
data.drop_duplicates()
# Now, those values if exists would have been dropped
```

Out[17]:

	Manufacturer	Model	Sales_in_thousands	__year_resale_value	Vehicle_type	Price_in_thousands	Engine_size	Horsepower	Wheelbase	Wid
0	Acura	Integra	16.919	16.360	Passenger	21.50	1.8	140.0	101.2	67
1	Acura	TL	39.384	19.875	Passenger	28.40	3.2	225.0	108.1	70
2	Acura	CL	14.114	18.225	Passenger	NaN	3.2	225.0	106.9	70
3	Acura	RL	8.588	29.725	Passenger	42.00	3.5	210.0	114.6	71
4	Audi	A4	20.397	22.255	Passenger	23.99	1.8	150.0	102.6	68
...	...	...	...	...	...	...	...	...	...	
152	Volvo	V40	3.545	NaN	Passenger	24.40	1.9	160.0	100.5	67
153	Volvo	S70	15.245	NaN	Passenger	27.50	2.4	168.0	104.9	69
154	Volvo	V70	17.531	NaN	Passenger	28.80	2.4	168.0	104.9	69
155	Volvo	C70	3.493	NaN	Passenger	45.50	2.3	236.0	104.9	71
156	Volvo	S80	18.969	NaN	Passenger	36.00	2.9	201.0	109.9	72

157 rows × 16 columns

In [18]:
```
# Here in this dataset, we do not have any duplicates as checked above but if we had then by using the below synt
# data.drop_duplicates(subset='Column_name',inplace=False)
```

5. Handling of Outliers

In [19]:
```
e0=data.Engine_size.min()
e100=data.Engine_size.max()
q1=data.Engine_size.quantile(0.25)
q2=data.Engine_size.quantile(0.5)
q3=data.Engine_size.quantile(0.75)
IQR=q3-q1
# IQR - Inter Quartile Range
# q1 - First_quartile
# q2 - Second_quatile
# q3 - Third_quartile
```

In [20]:
```
lf = q1 - 1.5*IQR
# lc - lower_fence
uf = q3 + 1.5*IQR
# uf - upper_fence
```

In [21]:
```
lf
```

Out[21]: 0.3874999999999993

In [22]:
```
uf
```

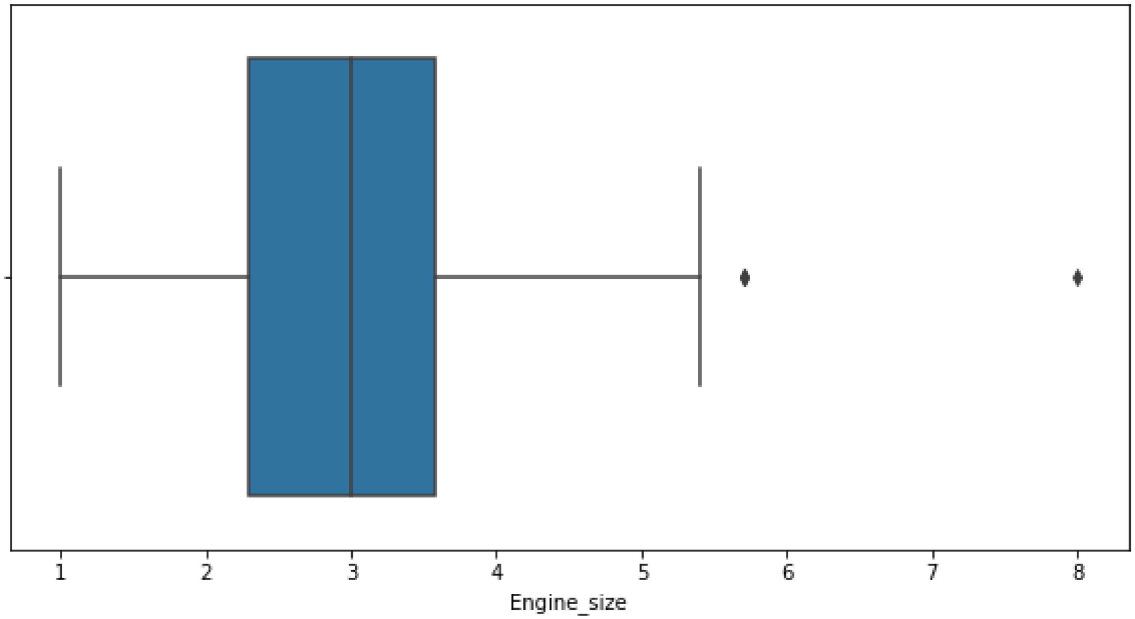
Out[22]: 5.487500000000001

In [23]:
```
# If lf<e0 then there are NO Outliers on the lower side
# If uf>e100 then there are NO Outliers on the higher side
```

In [24]:
```
print( "e0 = " , e0 ,", e100 = " , e100 ,", lf = " , lf ,", uf = " , uf)
```

```
e0 =  1.0 , e100 =  8.0 , lf =  0.3874999999999993 , uf =  5.487500000000001
```

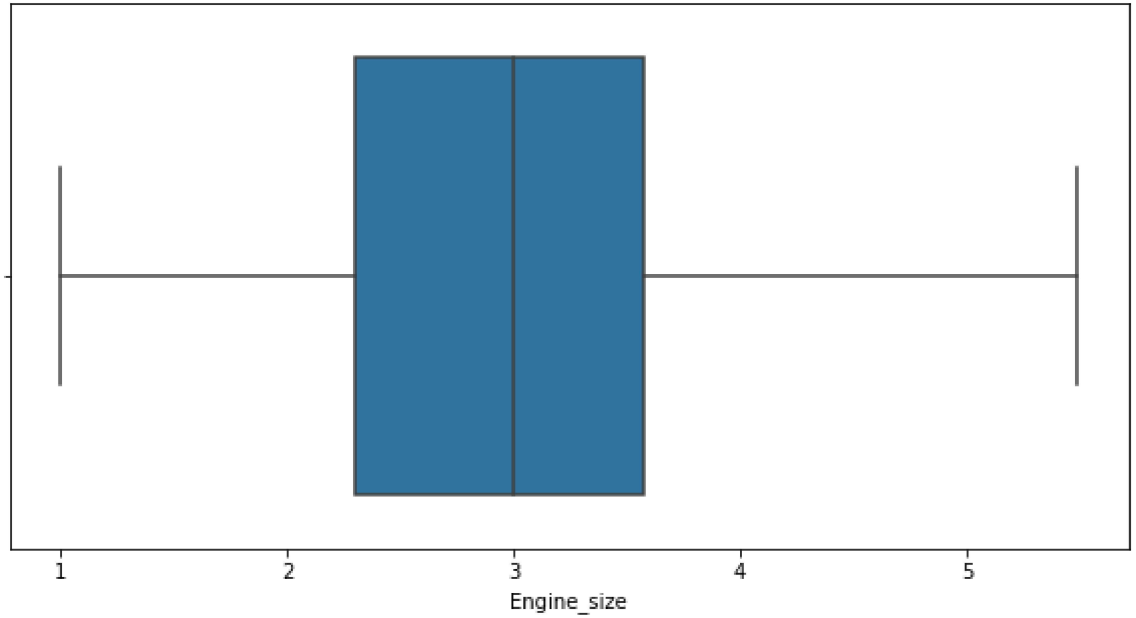
In [25]:
```
# Let's check the outliers in 'Engine_size' of this Dataset using Boxplot
plt.figure(figsize=(10,5))
sns.boxplot(data=data,x="Engine_size")
plt.show()
```



In [26]:
```
data.Engine_size.clip(upper=uf)
```

Out[26]:
```
0      1.8
1      3.2
2      3.2
3      3.5
4      1.8
      ...
152    1.9
153    2.4
154    2.4
155    2.3
156    2.9
Name: Engine_size, Length: 157, dtype: float64
```

In [27]:
```
data.Engine_size.clip(upper=uf,inplace=True)
plt.figure(figsize=(10,5))
sns.boxplot(data=data,x="Engine_size")
plt.show()
# So, In this we can see that the outliers are dropped out of the dataset and new Min and Max values have been as
```



6. Handle Missing Values

In [28]:
```
# Detecting the Missing Values
data.__year_resale_value=pd.to_numeric(data.__year_resale_value,errors='coerce')
data.isnull().sum()
# Dataset.isnull.sum() will give the number of missing values in the dataset
# error='coerce' will replace the non-numeric values with NaN
```

Out[28]:
```
Manufacturer            0
Model                   0
Sales_in_thousands      0
__year_resale_value    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
Power_perf_factor       2
dtype: int64
```

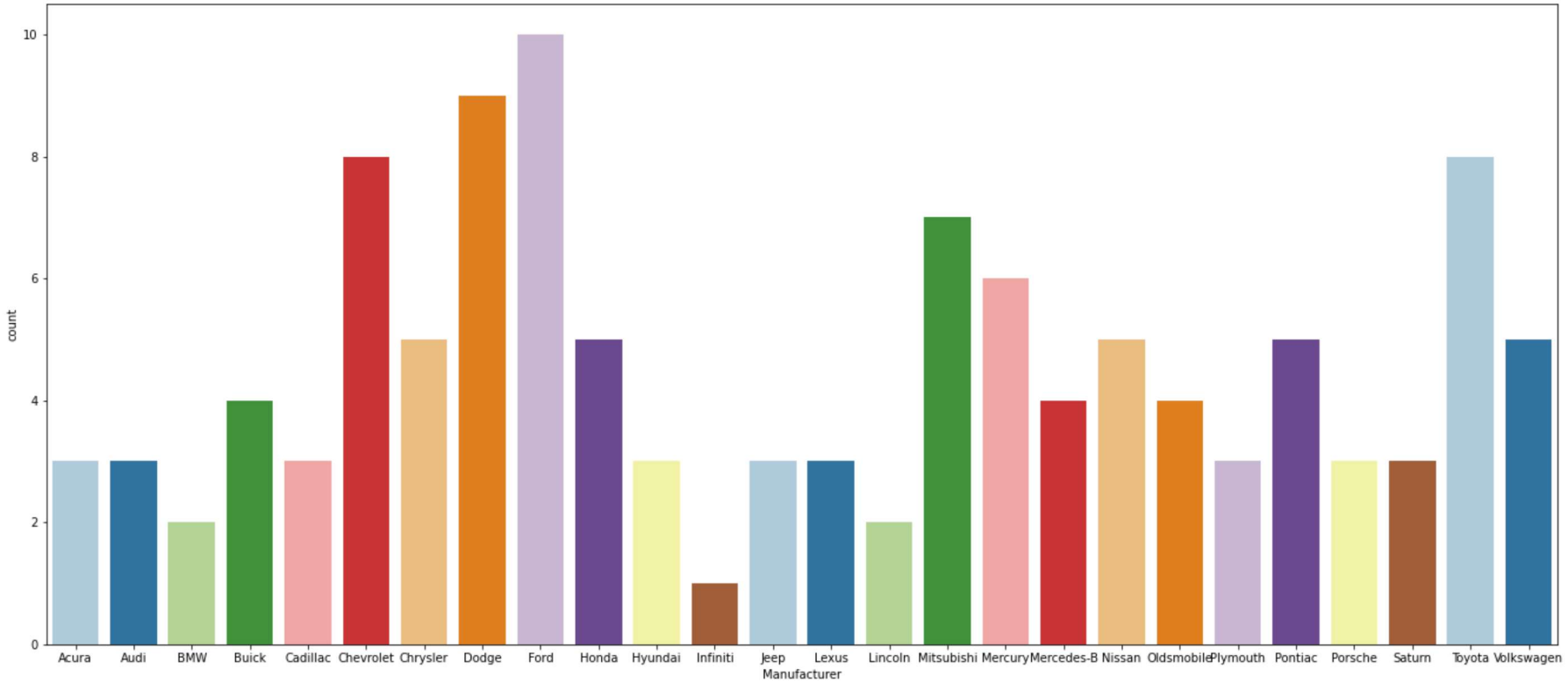
In [29]:
```
# Missing Values in the Dataset
# __year_resale_value = 36
# Fuel_efficieny = 3
# all other attributes also have some of the missing values

data.dropna(how='any',inplace=True)
# After this all the missing values are dropped from the dataset
```

In [30]:
```
data.isnull().sum()
# Here we can check that all the null value counts are zero
```

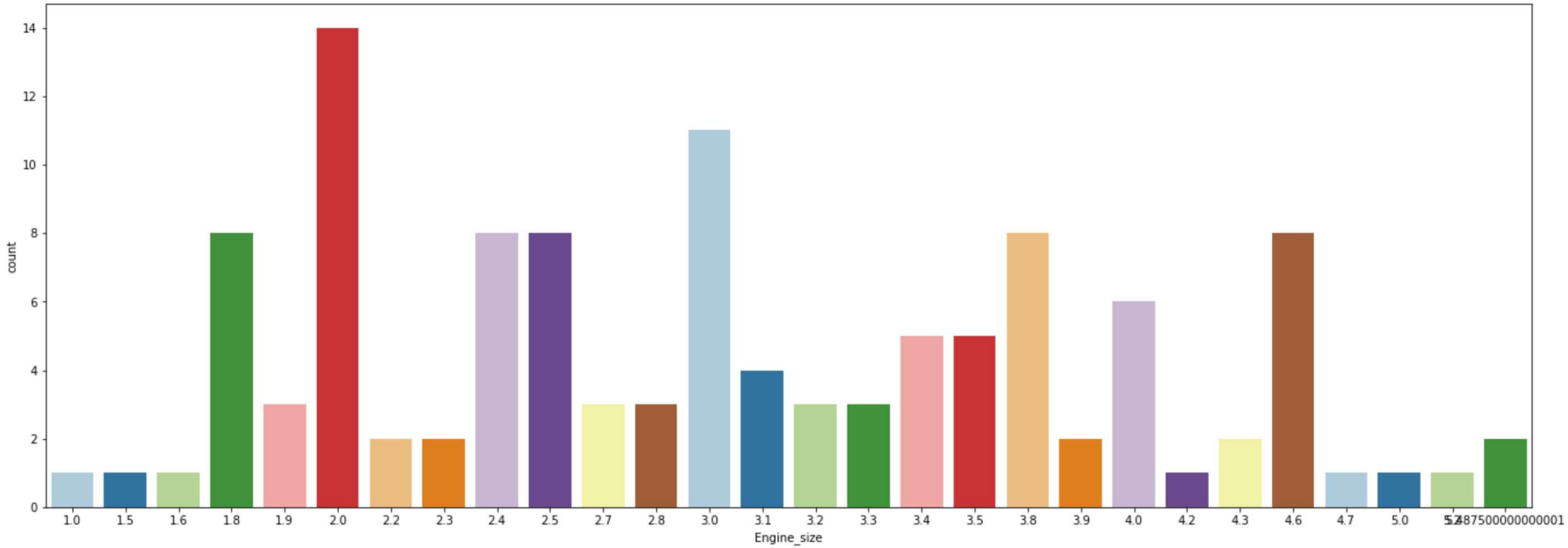
Out[30]:
```
Manufacturer          0
Model                 0
Sales_in_thousands    0
__year_resale_value   0
Vehicle_type          0
Price_in_thousands    0
Engine_size           0
Horsepower            0
Wheelbase             0
Width                 0
Length                0
Curb_weight           0
Fuel_capacity         0
Fuel_efficiency       0
Latest_Launch         0
Power_perf_factor     0
dtype: int64
```

7. Univariate Analysis (Taking only one attribute out of Data)

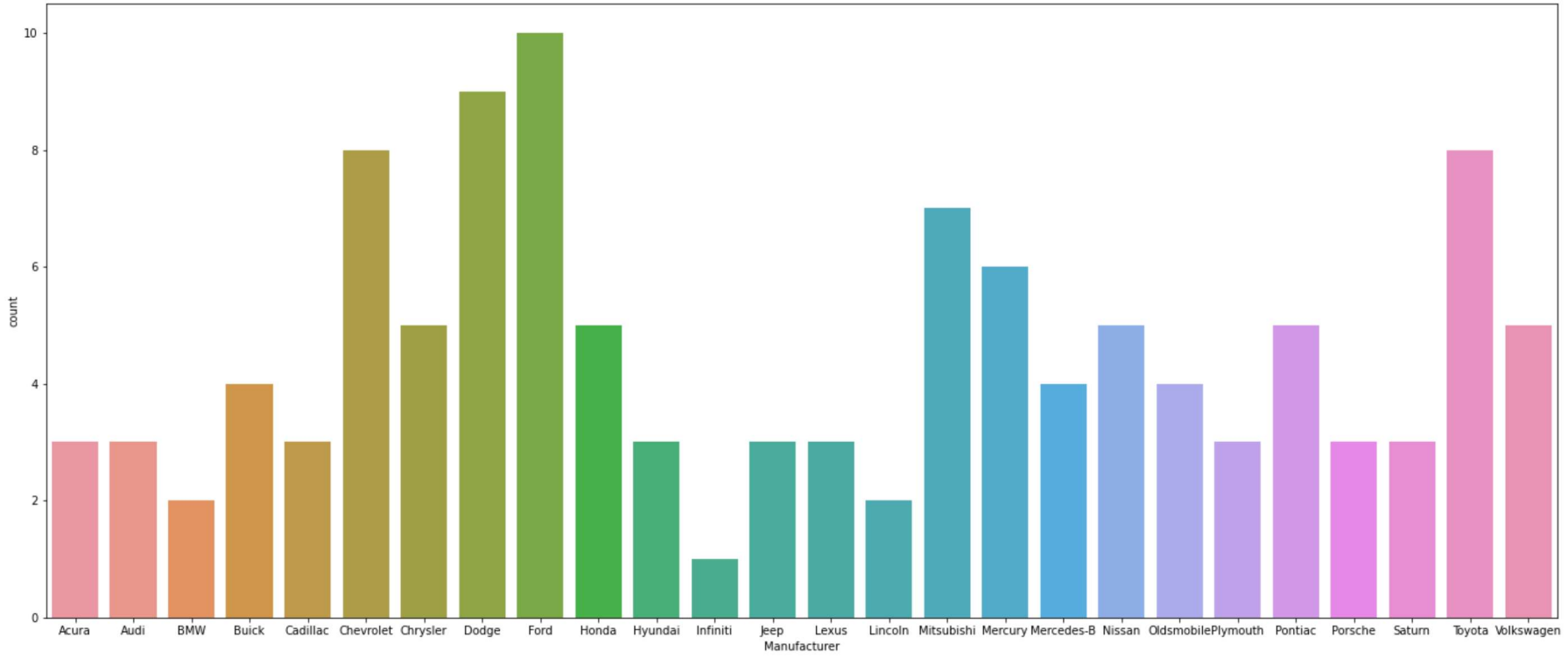
In [31]:
```
plt.figure(figsize=(23,10))
sns.countplot(x=data['Manufacturer'],palette='Paired')
plt.show()
# CountPlot will basically display the count of number of the cars made by the Manufacturer
```



In [32]:
```
# Engine type is measured in Litres(1l to 8l)
plt.figure(figsize=(23,8))
sns.countplot(x=data['Engine_size'],palette='Paired')
plt.show()
```

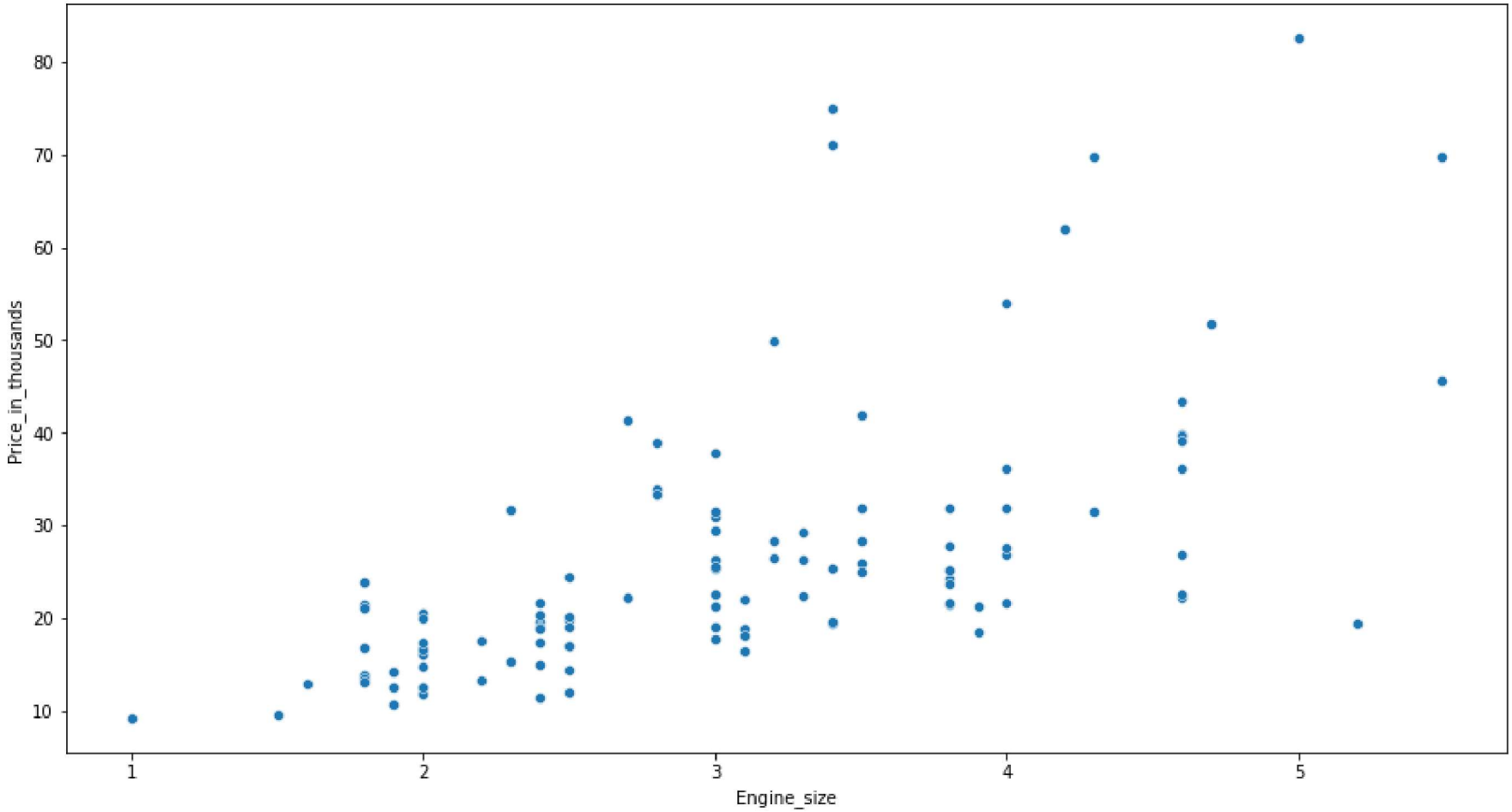


In [33]:
```
plt.figure(figsize=(24,10))
sns.countplot(x='Manufacturer',data=data)
plt.show()
# This plot is showing a lot of cars of different Manufacterers below with their respective count
```

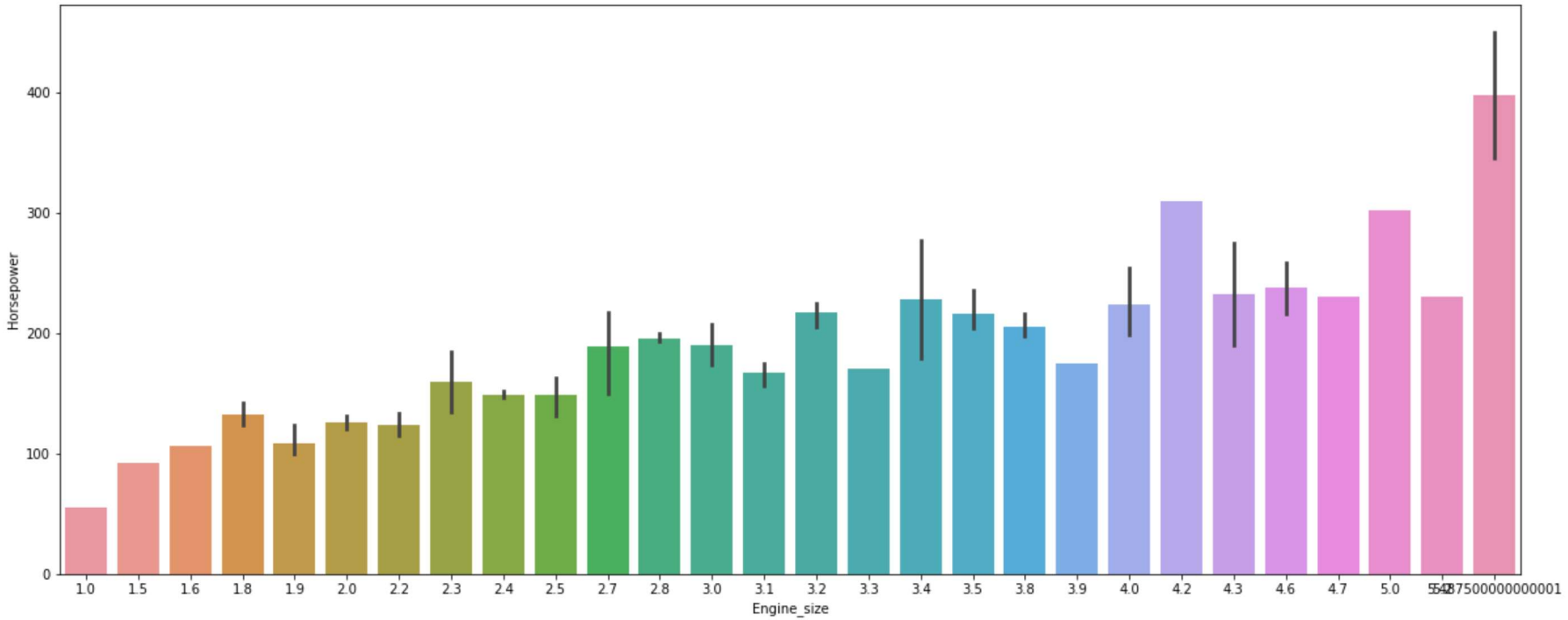


8. Bivariate Analysis (Taking only two attribute out of Data)

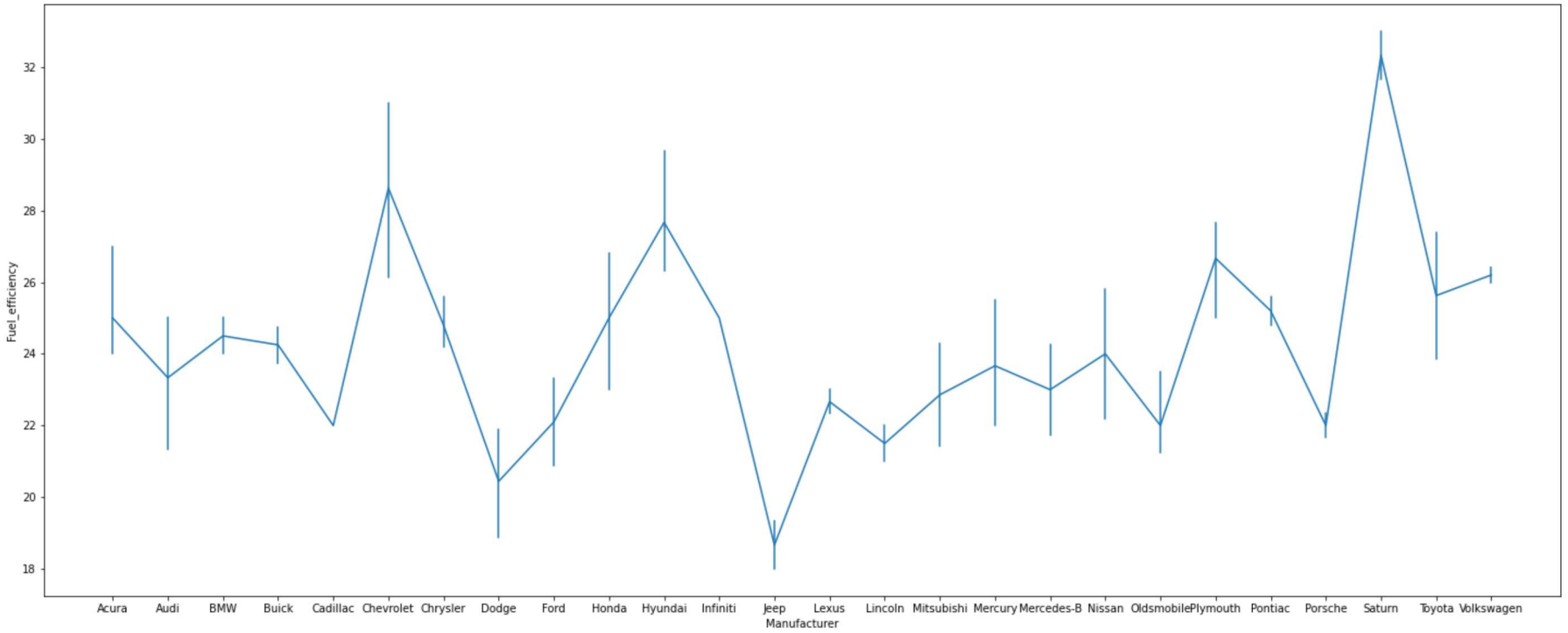
In [34]:
```
# Scatter Plot of 'Engine_size' verse 'Price_of_the_Vehicles'
plt.figure(figsize=(15,8))
sns.scatterplot(data=data,x="Engine_size",y='Price_in_thousands')
plt.show()
```



In [35]:
```
plt.figure(figsize=(20,8))
sns.barplot(data=data,x='Engine_size',y='Horsepower')
plt.show()
#This barplot is showing 'Horsepower' verse 'Engine_size' of the cars
```



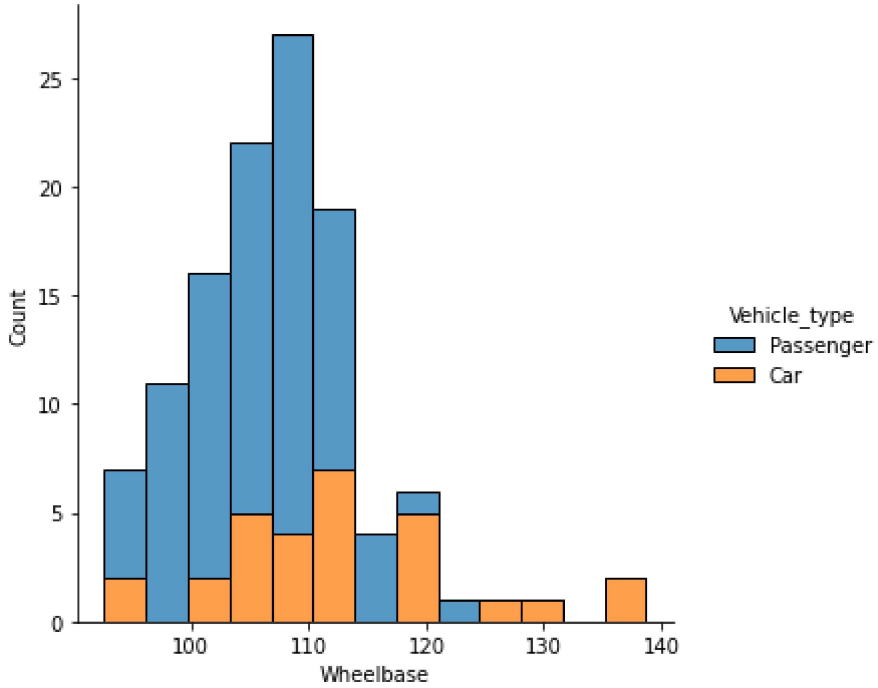
In [36]:
```
plt.figure(figsize=(25,10))
sns.lineplot(
    data=data, x="Manufacturer", y="Fuel_efficiency", err_style="bars", ci=68
)
plt.show()

# So in this lineplot, we will the 'fuel_efficiency' verse 'Manufacturer' in 'bars' style mode with 'confidence_i
```



In [37]:
```
# In above lineplot we can see that high fuel_efficiency is give by 'Saturn' cars and unfortunately this car comp
# After that 'Chevrolet' cars are performing good and after that 'Hyundai'
# These bars are telling about the 'fuel_efficiency' range of each car manufacturer.
```

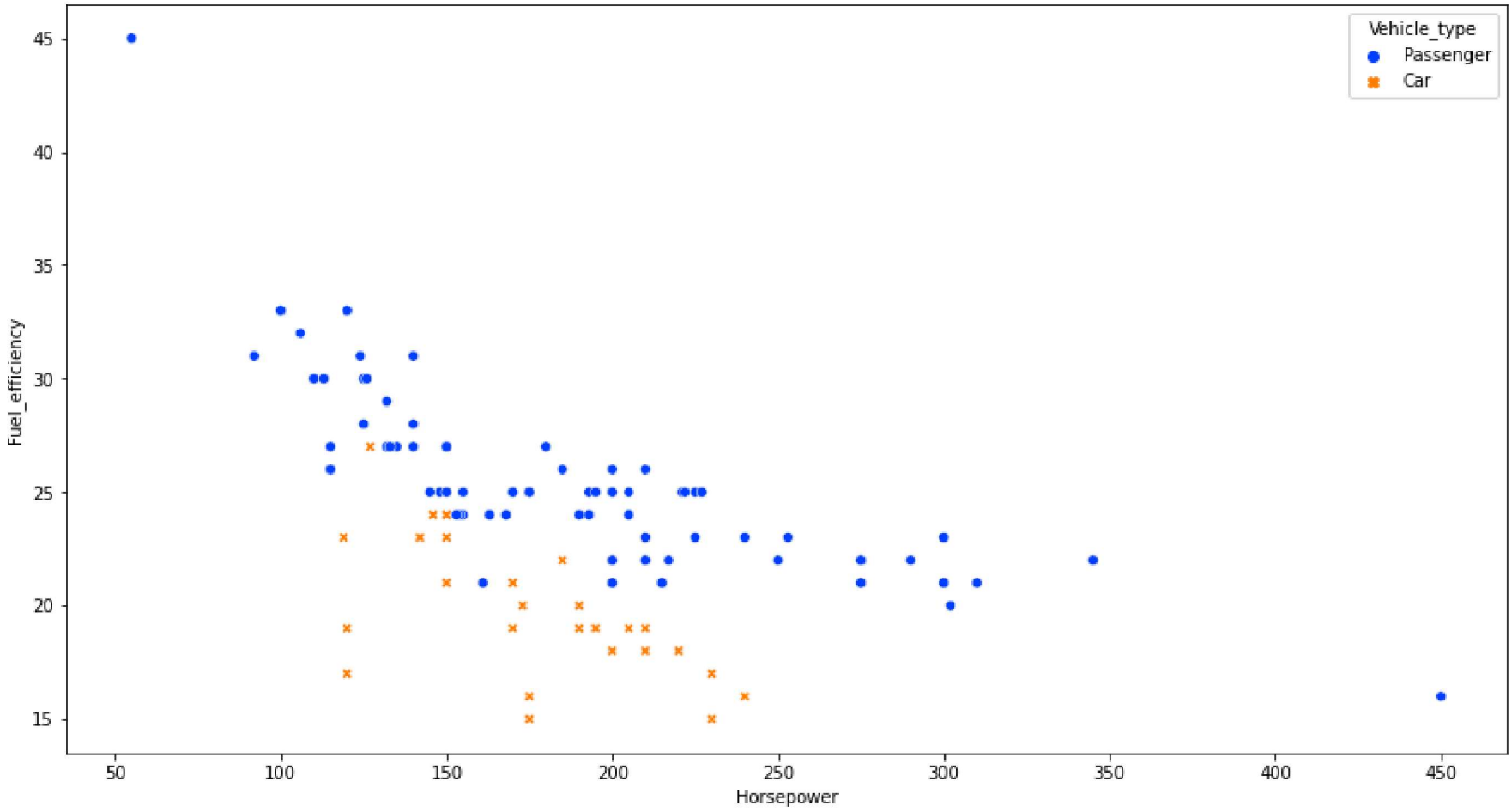
In [38]:
```
sns.displot(data=data, x="Wheelbase", hue="Vehicle_type", multiple="stack")
# Here actually we are trying to make a distplot using Seaborn after giving the 'Vehicle_type' for a bit of color
```

Out[38]: <seaborn.axisgrid.FacetGrid at 0x19de4d82490>

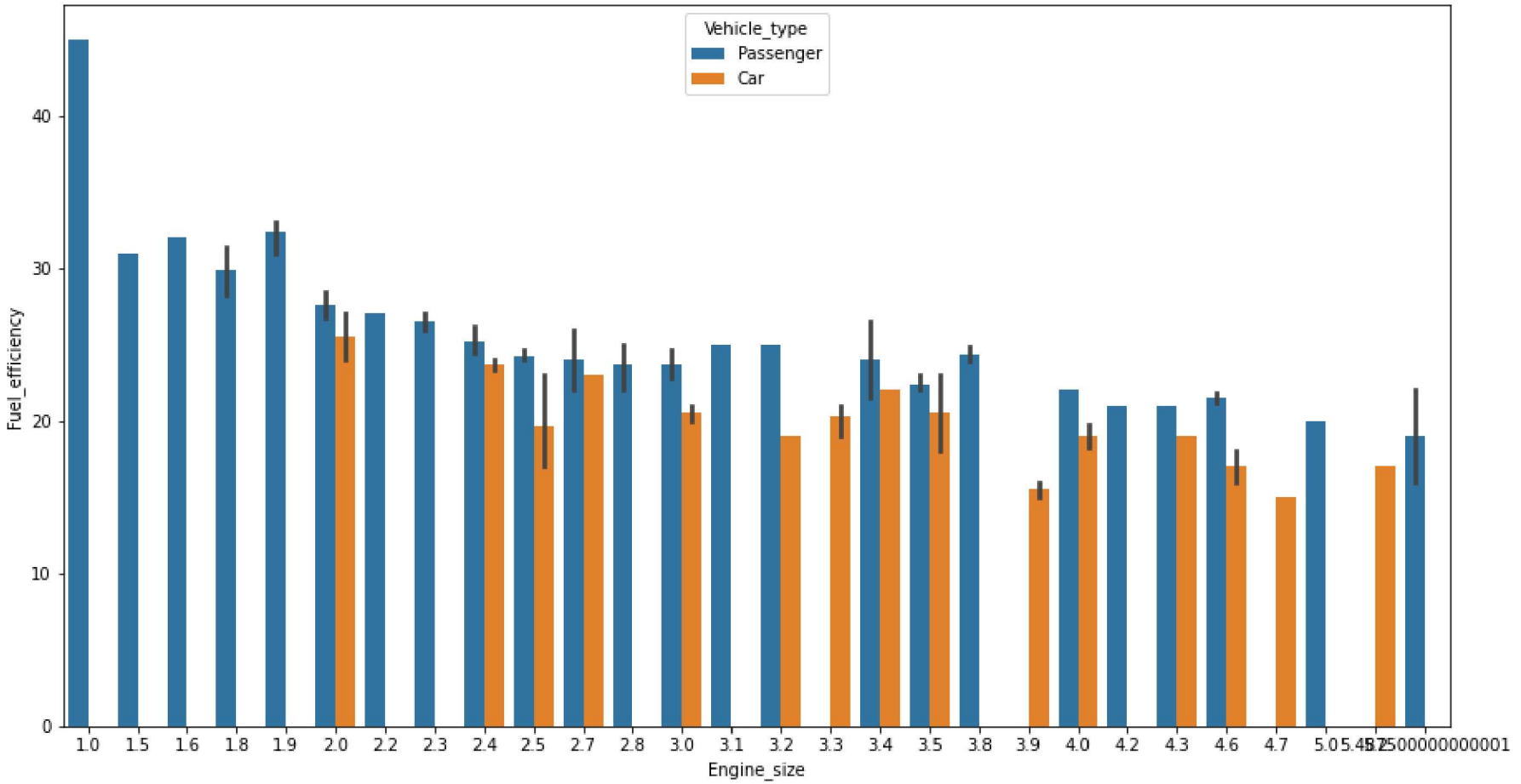


9. Multivariate Analysis (Taking multiple attribute out of Data)

In [39]:
```
plt.figure(figsize=(15,8))
sns.scatterplot(data=data,x='Horsepower',y='Fuel_efficiency',palette='bright',hue='Vehicle_type',style='Vehicle_t
plt.show()
# This scatterplot is showing the visualization between the different attributes of the given Data with 'Vehicle_
```



In [40]:
```
plt.figure(figsize=(15,8))
sns.barplot(data=data,x='Engine_size',y='Fuel_efficiency',hue='Vehicle_type')
plt.show()
# Here a Bar plot with three attributes have been visualized to the in bound relations between them.
```



In [41]:
```
sns.pairplot(data=data, vars=['Sales_in_thousands','__year_resale_value','Price_in_thousands'],hue='Vehicle_type'
# In this pairplot, we can actually see a mixer of different plots such as Density (KDE) plot, Scatterplot,histog
```

Out[41]: <seaborn.axisgrid.PairGrid at 0x19dde7ff4f0>