Import Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
from scipy import stats
from scipy.stats import skew
from sklearn.preprocessing import MinMaxScaler
from scipy.stats import zscore
import warnings
warnings.filterwarnings("ignore")
```

Read Car_Sales.csv Data

In [2]:
```
df = pd.read_csv(r"C:\Users\north\OneDrive\Desktop\zzz\States\Car_sales.cs
```

Shape of Data

In [3]:
```
df.shape
```

Out[3]: (157, 16)

Let's Explore the data

In [4]:
```
df.head(10)
```

Out[4]:

	Manufacturer	Model	Sales_in_thousands	__year_resale_value	Vehicle_type	Price_in_thousar
0	Acura	Integra	16.919	16.360	Passenger	21
1	Acura	TL	39.384	19.875	Passenger	28
2	Acura	CL	14.114	18.225	Passenger	N
3	Acura	RL	8.588	29.725	Passenger	42
4	Audi	A4	20.397	22.255	Passenger	23
5	Audi	A6	18.780	23.555	Passenger	33
6	Audi	A8	1.380	39.000	Passenger	62
7	BMW	323i	19.747	NaN	Passenger	26
8	BMW	328i	9.231	28.675	Passenger	33
9	BMW	528i	17.527	36.125	Passenger	38

In [5]:
```
1 df.tail(10)
```

Out[5]:

	Manufacturer	Model	Sales_in_thousands	__year_resale_value	Vehicle_type	Price_in_thous
147	Volkswagen	Passat	51.102	16.725	Passenger	
148	Volkswagen	Cabrio	9.569	16.575	Passenger	
149	Volkswagen	GTI	5.596	13.760	Passenger	
150	Volkswagen	Beetle	49.463	NaN	Passenger	
151	Volvo	S40	16.957	NaN	Passenger	
152	Volvo	V40	3.545	NaN	Passenger	
153	Volvo	S70	15.245	NaN	Passenger	
154	Volvo	V70	17.531	NaN	Passenger	
155	Volvo	C70	3.493	NaN	Passenger	
156	Volvo	S80	18.969	NaN	Passenger	

In [6]:
```
1 df.columns = [col.replace('_', ' ')
2 for col in df.columns]
```

In [7]:
```
1 df.columns = df.columns.str.strip()
```

In [8]:
```
1 df.columns
```

Out[8]:
```
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')
```

Basic information about the dataset and data types

In [9]:
```
1 df.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 [10]:
```
1 df.dtypes
```

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
```

Missing values

Prior to summary statistics, identify missing values or duplicates and drop them.

In [11]:
```
df.isna().sum()
```

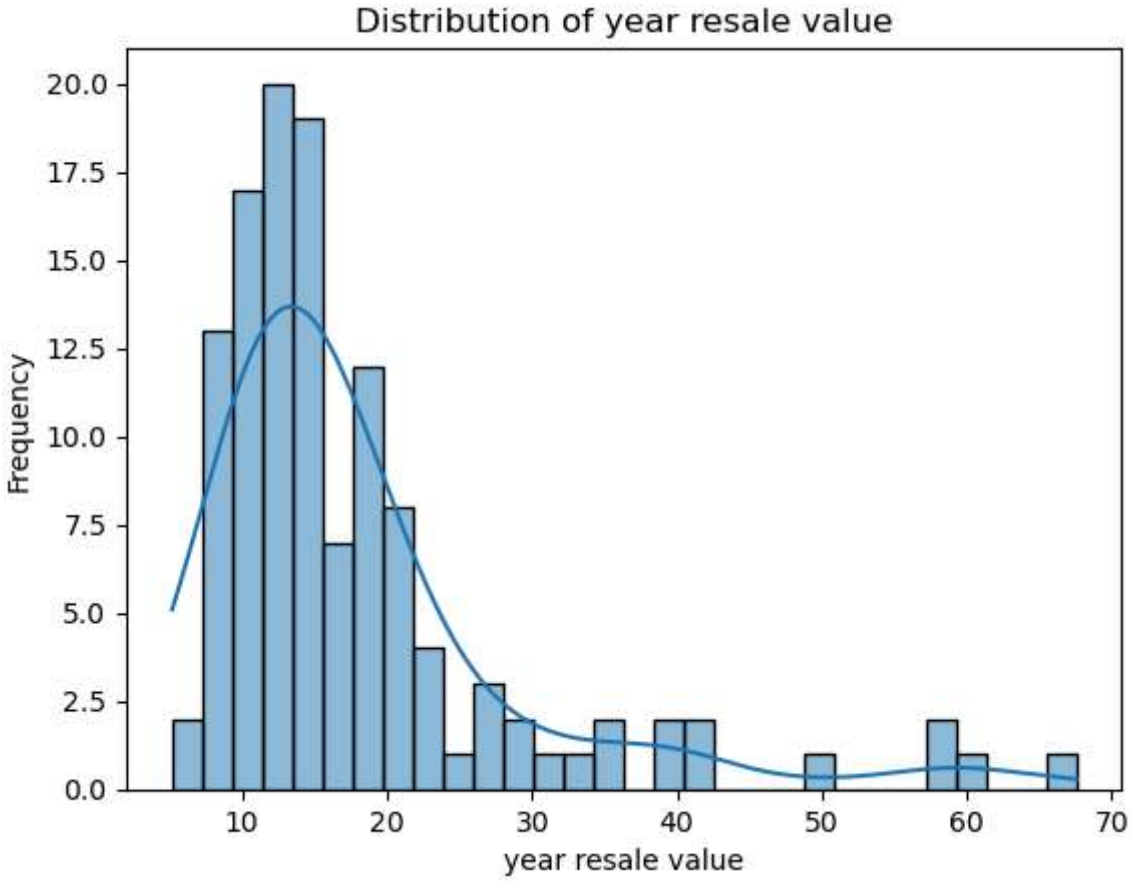
Out[11]:
```
Manufacturer          0
Model                 0
Sales in thousands    0
year resale value    36
Vehicle type          0
Price in thousands    2
Engine size           1
Horsepower            1
Wheelbase             1
Width                 1
Length                1
Curb weight           2
Fuel capacity         1
Fuel efficiency       3
Latest Launch         0
Power perf factor     2
dtype: int64
```

Imputeing median value in missing

In [12]:
```
impute_col = ['Price in thousands', 'Engine size', 'Horsepower', 'Wheelbas
                    'Length', 'Curb weight', 'Fuel capacity', 'Fuel effic
for col in impute_col:
    df[col].fillna(df[col].median(), inplace=True)

```

year resale value

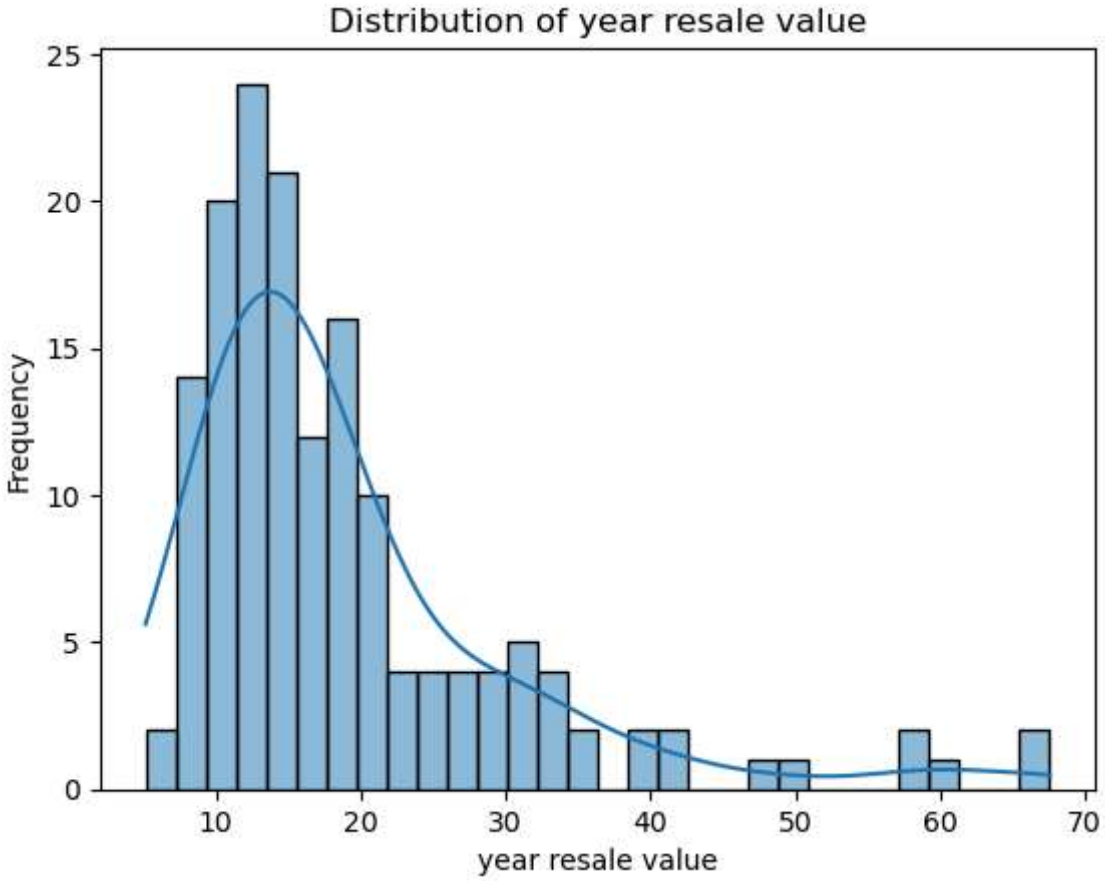
In [13]:
```
sns.histplot(df['year resale value'], bins=30, kde=True)
plt.title('Distribution of year resale value')
plt.xlabel('year resale value')
plt.ylabel('Frequency')
plt.show()
```



There are 36 missing values in year resale value, Which is 22% of the dataset. We can use Linear Regression to predict the missing values.

In [14]:
```
from sklearn.linear_model import LinearRegression

data_values = df[df['year resale value'].notna()]
missing_data = df[df['year resale value'].isna()]

model = LinearRegression()
model.fit(data_values[['Price in thousands', 'Horsepower']], data_values[

predicted_resale = model.predict(missing_data[['Price in thousands', 'Hors
df.loc[df['year resale value'].isna(), 'year resale value'] = predicted_re
```

In [15]:
```
sns.histplot(df['year resale value'], bins=30, kde=True)
plt.title('Distribution of year resale value')
plt.xlabel('year resale value')
plt.ylabel('Frequency')
plt.show()
```

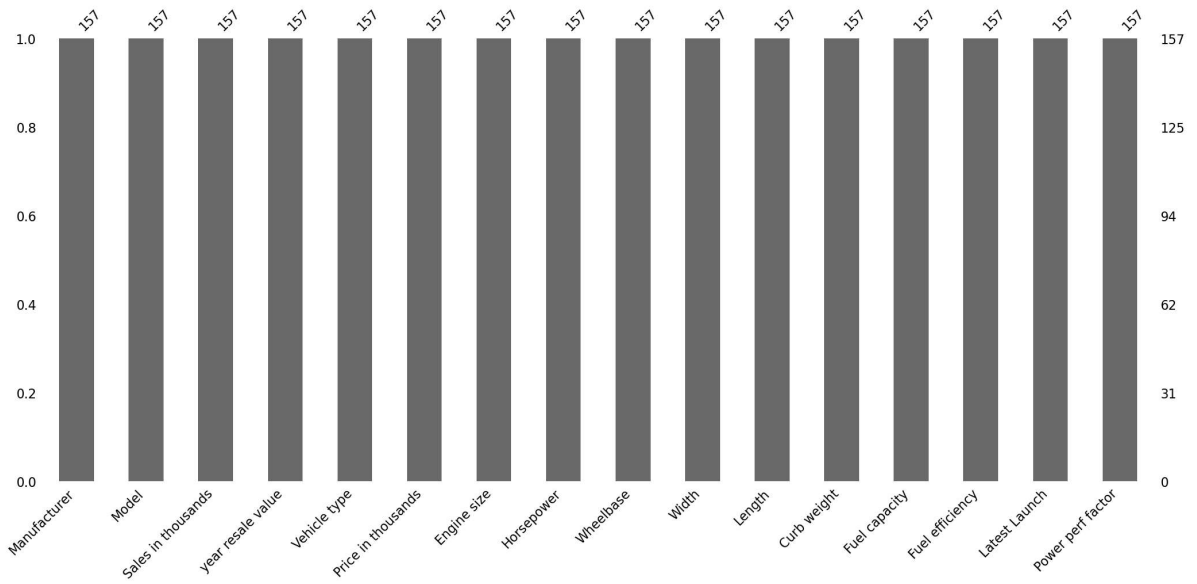


In [16]:
```
df.isna().sum()
```

Out[16]:
```
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
```

In [17]:
```
1 msno.bar(df)
```

Out[17]: <Axes: >



Check Duplicates

In [18]:
```
1 df.duplicated().sum()
```

Out[18]: 0

Keep only numeric columns in a dataframe

hint: (use function select_dtype)

In [19]:
```
1 df = df.select_dtypes(include=[np.number])
2 df.head()
```

Out[19]:

	Sales in thousands	year resale value	Price in thousands	Engine size	Horsepower	Wheelbase	Width	Length	Curb weight	I capa
0	16.919	16.360	21.500	1.8	140.0	101.2	67.3	172.4	2.639	
1	39.384	19.875	28.400	3.2	225.0	108.1	70.3	192.9	3.517	
2	14.114	18.225	22.799	3.2	225.0	106.9	70.6	192.0	3.470	
3	8.588	29.725	42.000	3.5	210.0	114.6	71.4	196.6	3.850	
4	20.397	22.255	23.990	1.8	150.0	102.6	68.2	178.0	2.998	

Summary statistics

In [20]:
```
1 df.describe().T
```

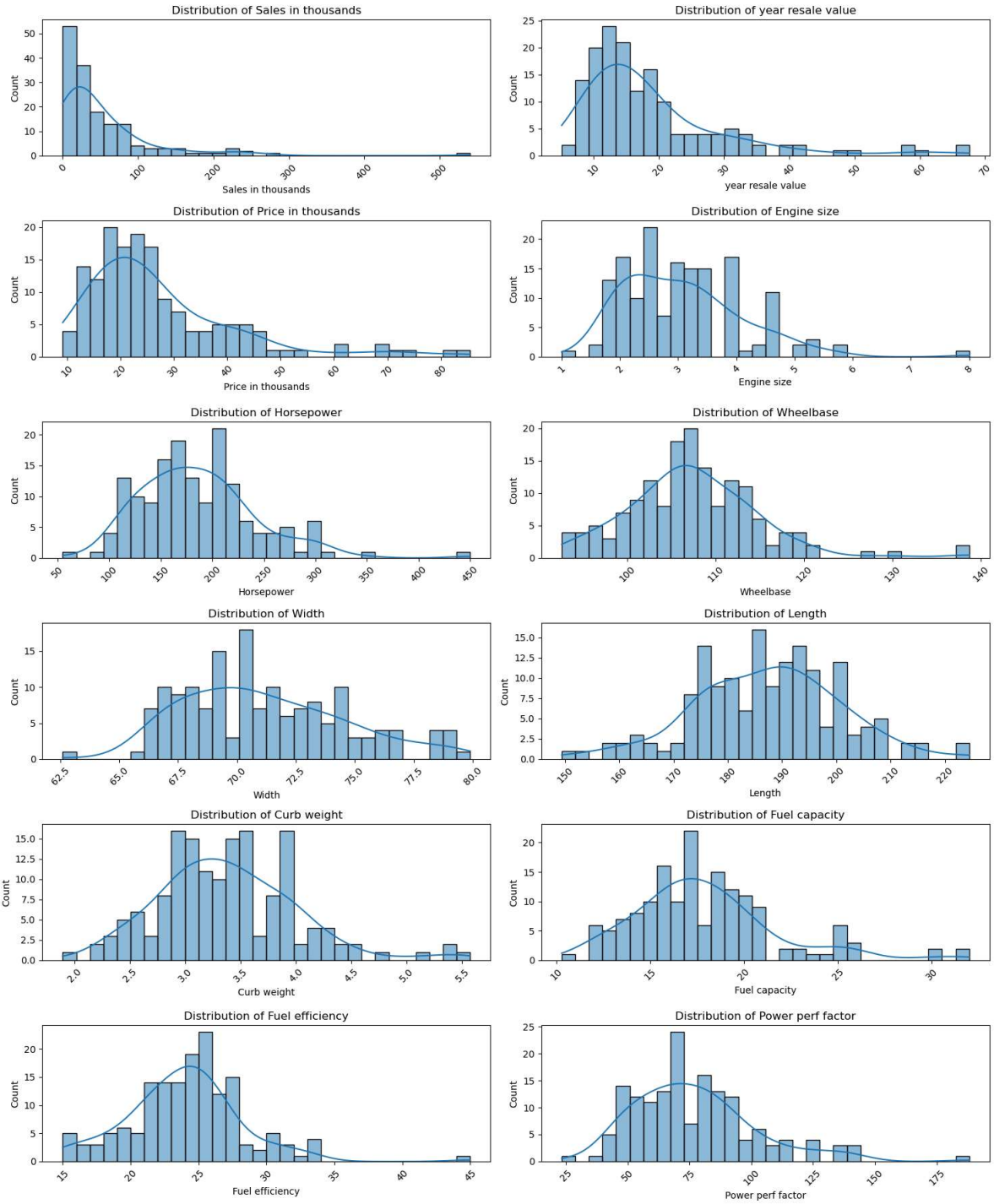
Out[20]:

	count	mean	std	min	25%	50%	75%	
Sales in thousands	157.0	52.998076	68.029422	0.110000	14.114000	29.450000	67.956000	540.
year resale value	157.0	19.114613	11.618144	5.160000	12.025000	15.434614	20.945114	67.
Price in thousands	157.0	27.332261	14.268713	9.235000	18.145000	22.799000	31.930000	85.
Engine size	157.0	3.060510	1.041311	1.000000	2.300000	3.000000	3.500000	8.
Horsepower	157.0	185.894904	56.522319	55.000000	150.000000	177.500000	215.000000	450.
Wheelbase	157.0	107.484076	7.616872	92.600000	103.000000	107.000000	112.200000	138.
Width	157.0	71.146178	3.441124	62.600000	68.400000	70.550000	73.400000	79.
Length	157.0	187.347134	13.388708	149.400000	177.600000	187.900000	196.100000	224.
Curb weight	157.0	3.377567	0.626460	1.895000	2.975000	3.342000	3.778000	5.
Fuel capacity	157.0	17.947134	3.875905	10.300000	15.800000	17.200000	19.500000	32.
Fuel efficiency	157.0	23.847134	4.241380	15.000000	21.000000	24.000000	26.000000	45.
Power perf factor	157.0	76.979735	24.987338	23.276272	60.727447	72.030917	89.401935	188.

Distribution

In [21]:

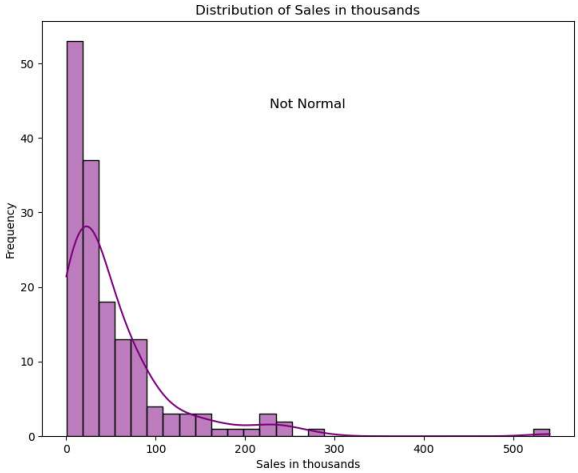
```
num_columns = len(df.columns)
num_rows = (num_columns + 1) // 2

plt.figure(figsize=(15, 3 * num_rows))
for i, column in enumerate(df.columns):
    plt.subplot(num_rows, 2, i + 1)
    sns.histplot(df[column], kde=True, bins=30)
    plt.title(f'Distribution of {column}')
    plt.xticks(rotation=45)

plt.tight_layout()
plt.show()
```

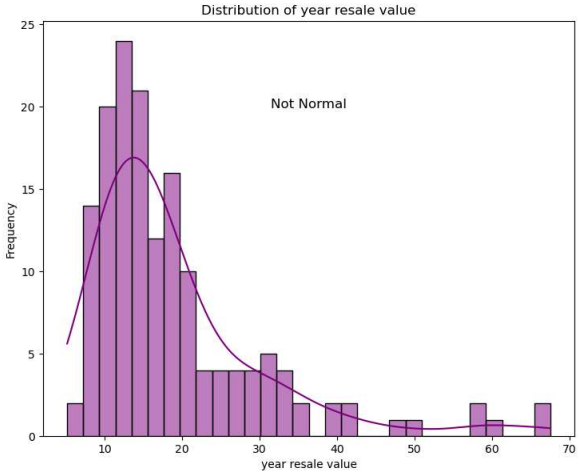

Distribution of Sales in thousands
Count
Sales in thousands
Distribution of year resale value
Count
year resale value
Distribution of Price in thousands
Count
Price in thousands
Distribution of Engine size
Count
Engine size
Distribution of Horsepower
Count
Horsepower
Distribution of Wheelbase
Count
Wheelbase
Distribution of Width
Count
Width
Distribution of Length
Count
Length
Distribution of Curb weight
Count
Curb weight
Distribution of Fuel capacity
Count
Fuel capacity
Distribution of Fuel efficiency
Count
Fuel efficiency
Distribution of Power perf factor
Count
Power perf factor

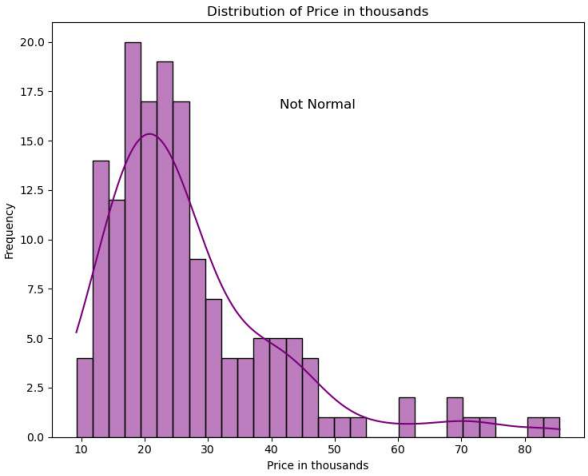
Normalized Distribution (Gaussian)

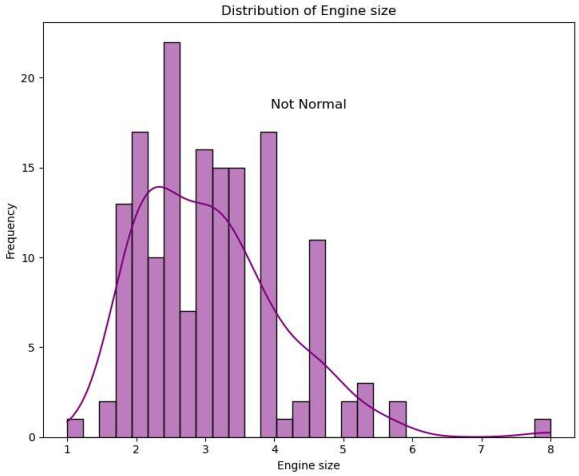
In [22]:

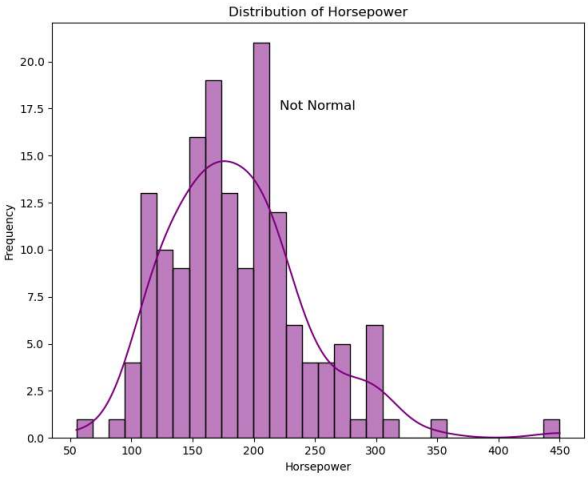
```
num_columns = len(df.columns)
num_rows = (num_columns + 1) // 2 if num_columns % 2 != 0 else num_columns

plt.figure(figsize=(15, 3 * len(df.columns)))

for i, col in enumerate(df.columns):
    normalized_data = df[col]
    k2, p = stats.normaltest(normalized_data)

    plt.subplot(num_rows, 2, i + 1)
    sns.histplot(normalized_data, bins=30, kde=True, color='purple')

    text = "Not Normal" if p < 0.05 else "Normal"
    plt.text(0.5, 0.8, text, horizontalalignment='center', verticalalignme
             transform=plt.gca().transAxes, fontsize=12)

    plt.title(f'Distribution of {col}')
    plt.xlabel(col)
    plt.ylabel('Frequency')

plt.tight_layout()
plt.show()
```

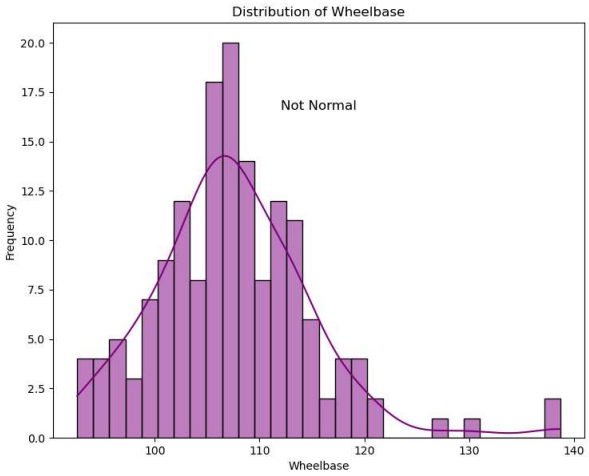

Distribution of Sales in thousands
Not Normal
Frequency
Sales in thousands

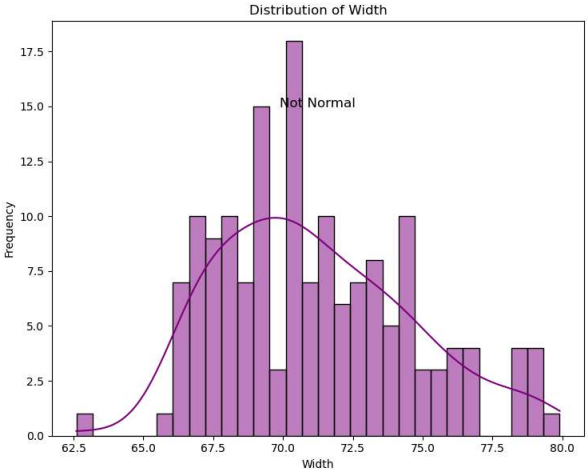

Distribution of year resale value
Not Normal
Frequency
year resale value

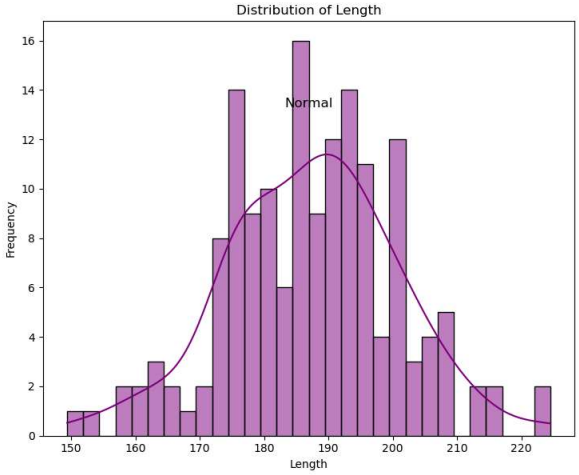

Distribution of Price in thousands
Not Normal
Frequency
Price in thousands

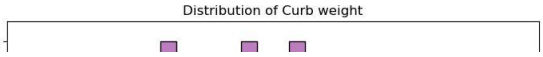

Distribution of Engine size
Not Normal
Frequency
Engine size

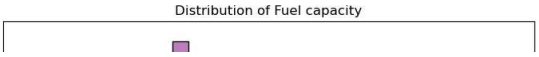

Distribution of Horsepower
Not Normal
Frequency
Horsepower

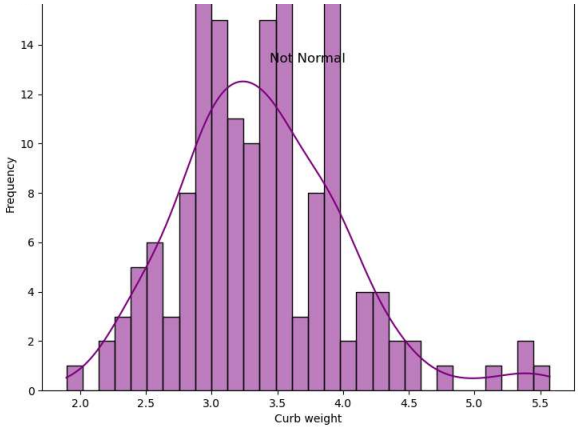

Distribution of Wheelbase
Not Normal
Frequency
Wheelbase

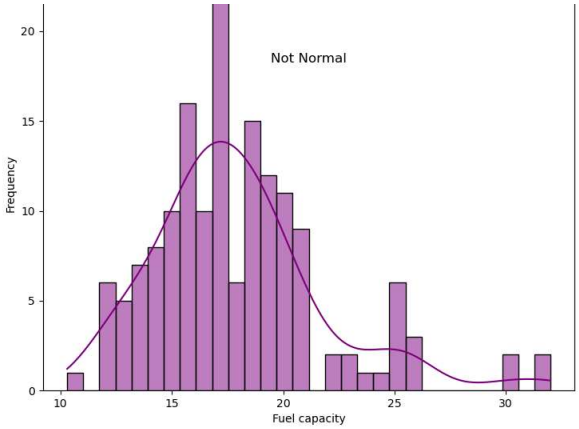

Distribution of Width
Not Normal
Frequency
Width

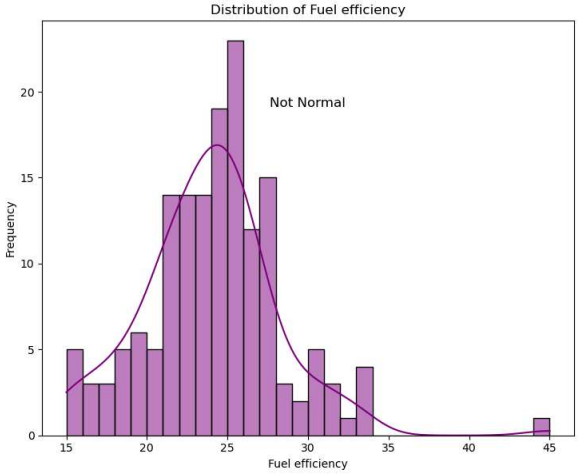

Distribution of Length
Normal
Frequency
Length

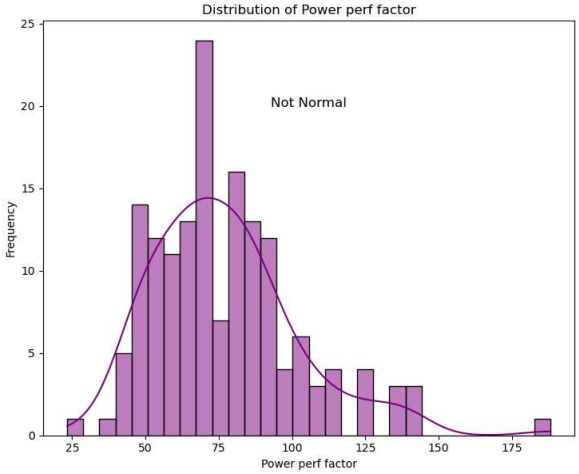

Distribution of Curb weight


Distribution of Fuel capacity


Not Normal
Frequency
Curb weight


Not Normal
Frequency
Fuel capacity

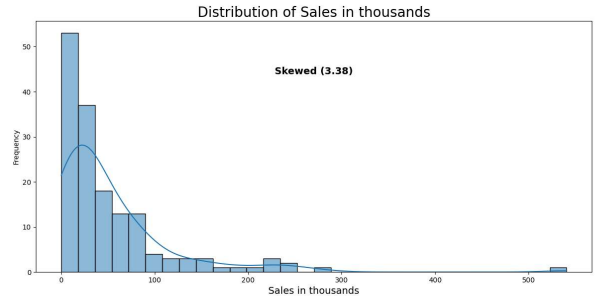

Distribution of Fuel efficiency
Not Normal
Frequency
Fuel efficiency

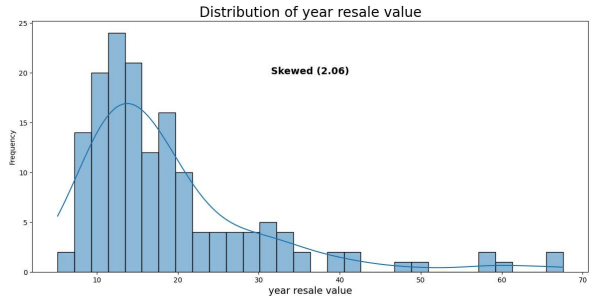

Distribution of Power perf factor
Not Normal
Frequency
Power perf factor

Skewed Distribution (negative and positive)

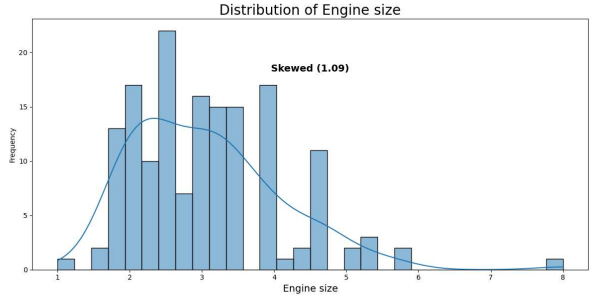
In [23]:

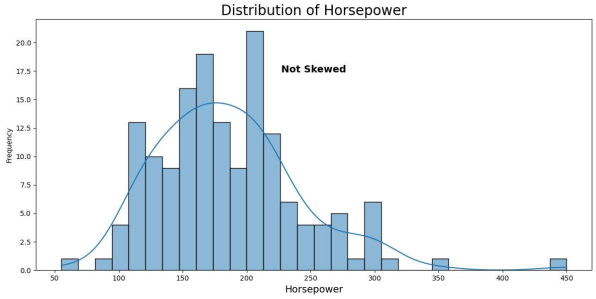
```
num_columns = len(df.columns)
num_rows = (num_columns + 1) // 2

plt.figure(figsize=(24, 6 * num_rows))

for i, column_name in enumerate(df.columns):
    col_data = df[column_name]
    skew_value = skew(col_data)

    plt.subplot(num_rows, 2, i + 1)
    sns.histplot(col_data, kde=True, bins=30)
    plt.title(f'Distribution of {column_name}', fontsize=20)
    plt.xlabel(column_name, fontsize=14)
    plt.ylabel('Frequency')

    skew_text = f"Skewed ({skew_value:.2f})" if skew_value < -1 or skew_va
    plt.text(0.5, 0.8, skew_text, horizontalalignment='center', verticalal

plt.tight_layout()
plt.show()
```

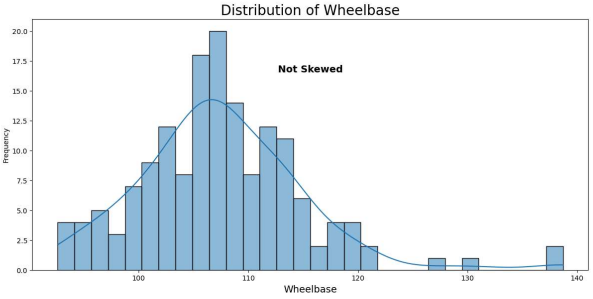

Distribution of Sales in thousands
Skewed (3.38)
Frequency
Sales in thousands

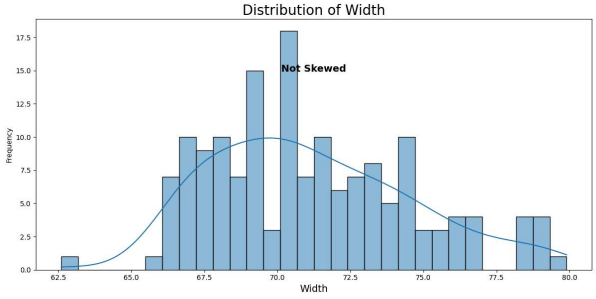

Distribution of year resale value
Skewed (2.06)
Frequency
year resale value

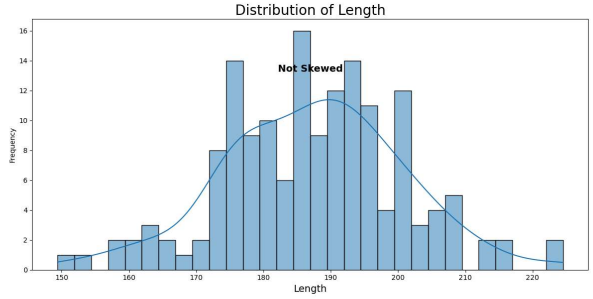

Distribution of Price in thousands
Skewed (1.77)
Frequency
Price in thousands

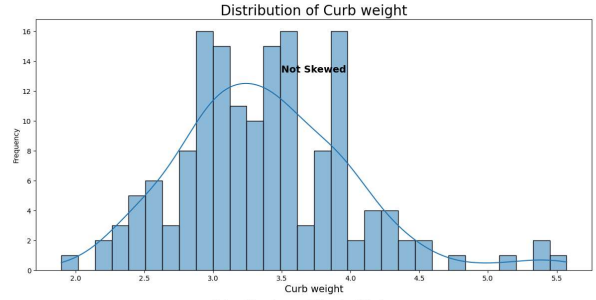

Distribution of Engine size
Skewed (1.09)
Frequency
Engine size

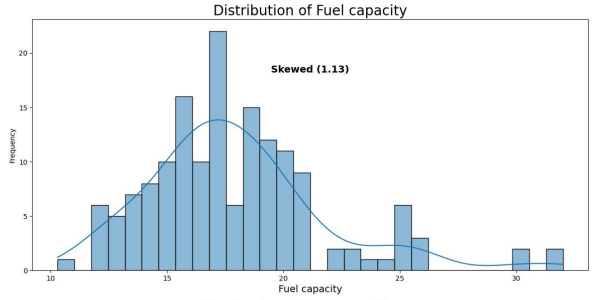

Distribution of Horsepower
Not Skewed
Frequency
Horsepower

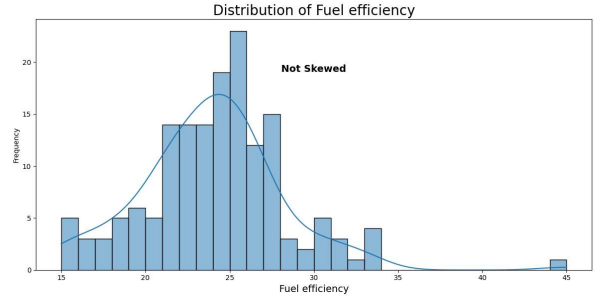

Distribution of Wheelbase
Not Skewed
Frequency
Wheelbase

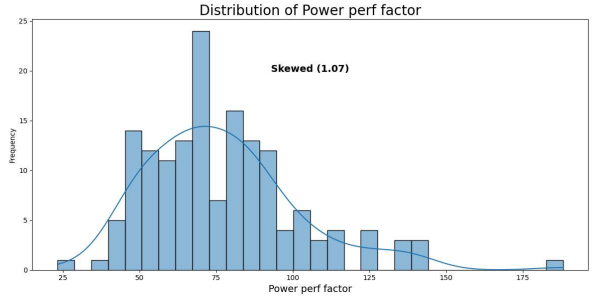

Distribution of Width
Not Skewed
Frequency
Width


Distribution of Length
Not Skewed
Frequency
Length


Distribution of Curb weight
Not Skewed
Frequency
Curb weight


Distribution of Fuel capacity
Skewed (1.13)
Frequency
Fuel capacity


Distribution of Fuel efficiency
Not Skewed
Frequency
Fuel efficiency

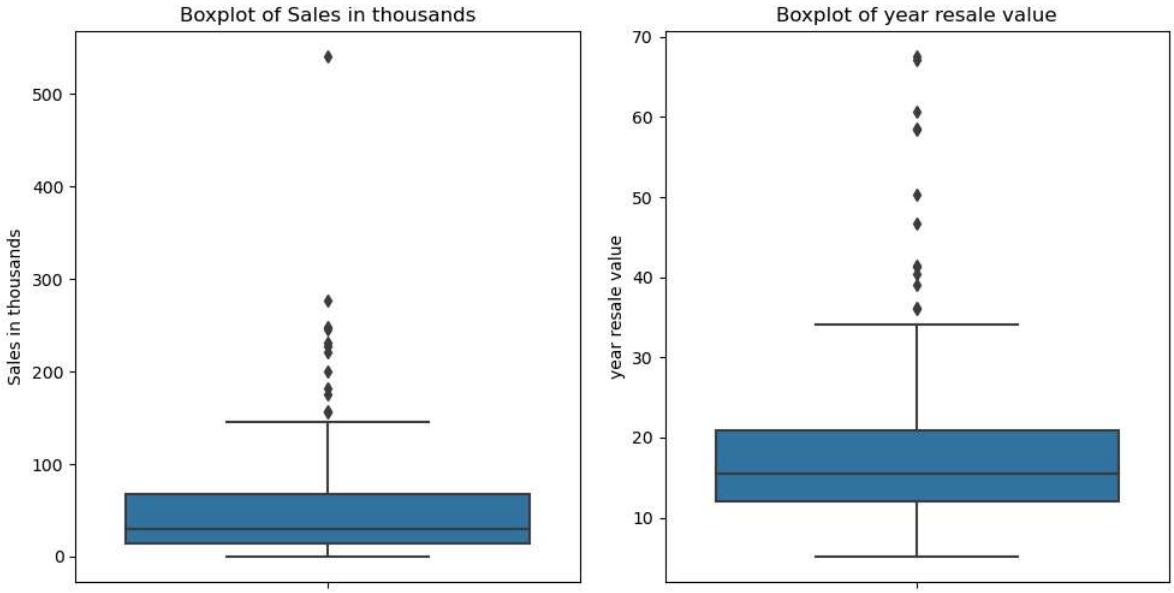

Distribution of Power perf factor
Skewed (1.07)
Frequency
Power perf factor

Outliers

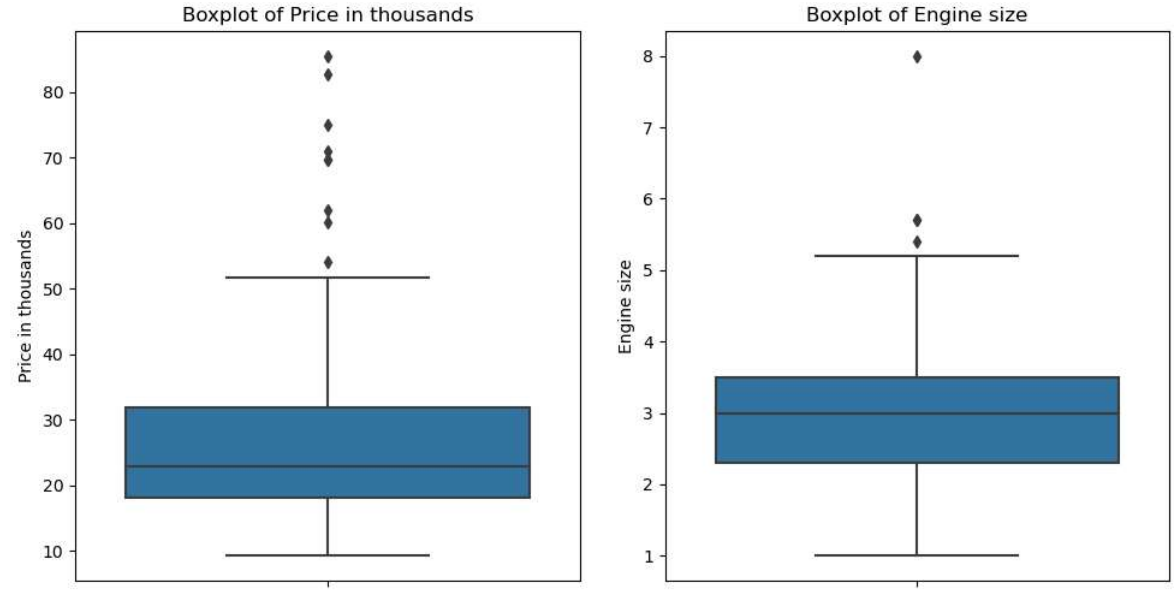
Plot Boxplot

In [24]:

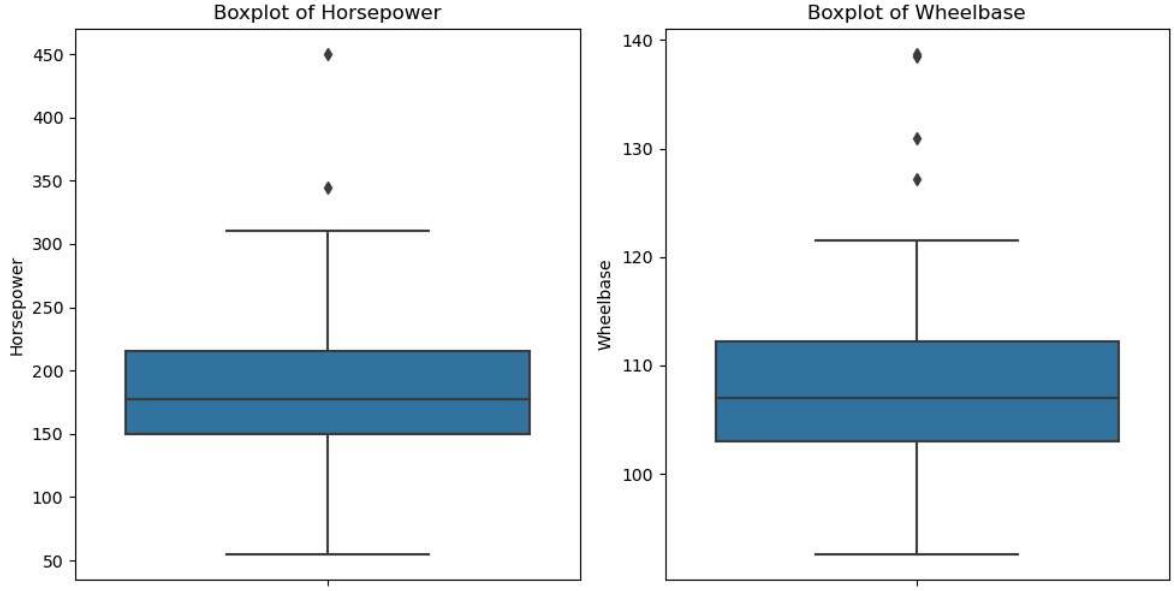
```
num_columns = len(df.columns)
num_rows = (num_columns + 1) // 2 if num_columns % 2 != 0 else num_columns

plt.figure(figsize=(10, 5 * num_rows))

for i, col in enumerate(df.columns):
    plt.subplot(num_rows, 2, i + 1)
    sns.boxplot(y=df[col])
    plt.title(f'Boxplot of {col}')
    plt.ylabel(col)

plt.tight_layout()
plt.show()
```


Boxplot of Sales in thousands
Sales in thousands
0
100
200
300
400
500

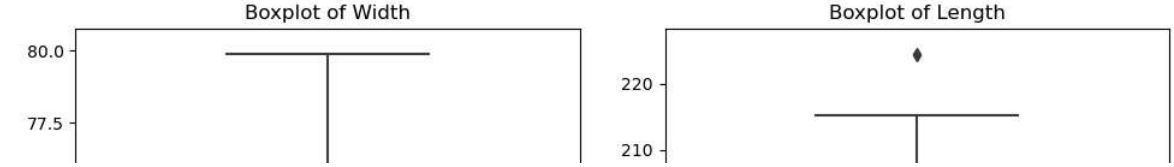
Boxplot of year resale value
year resale value
10
20
30
40
50
60
70


Boxplot of Price in thousands
Price in thousands
10
20
30
40
50
60
70
80

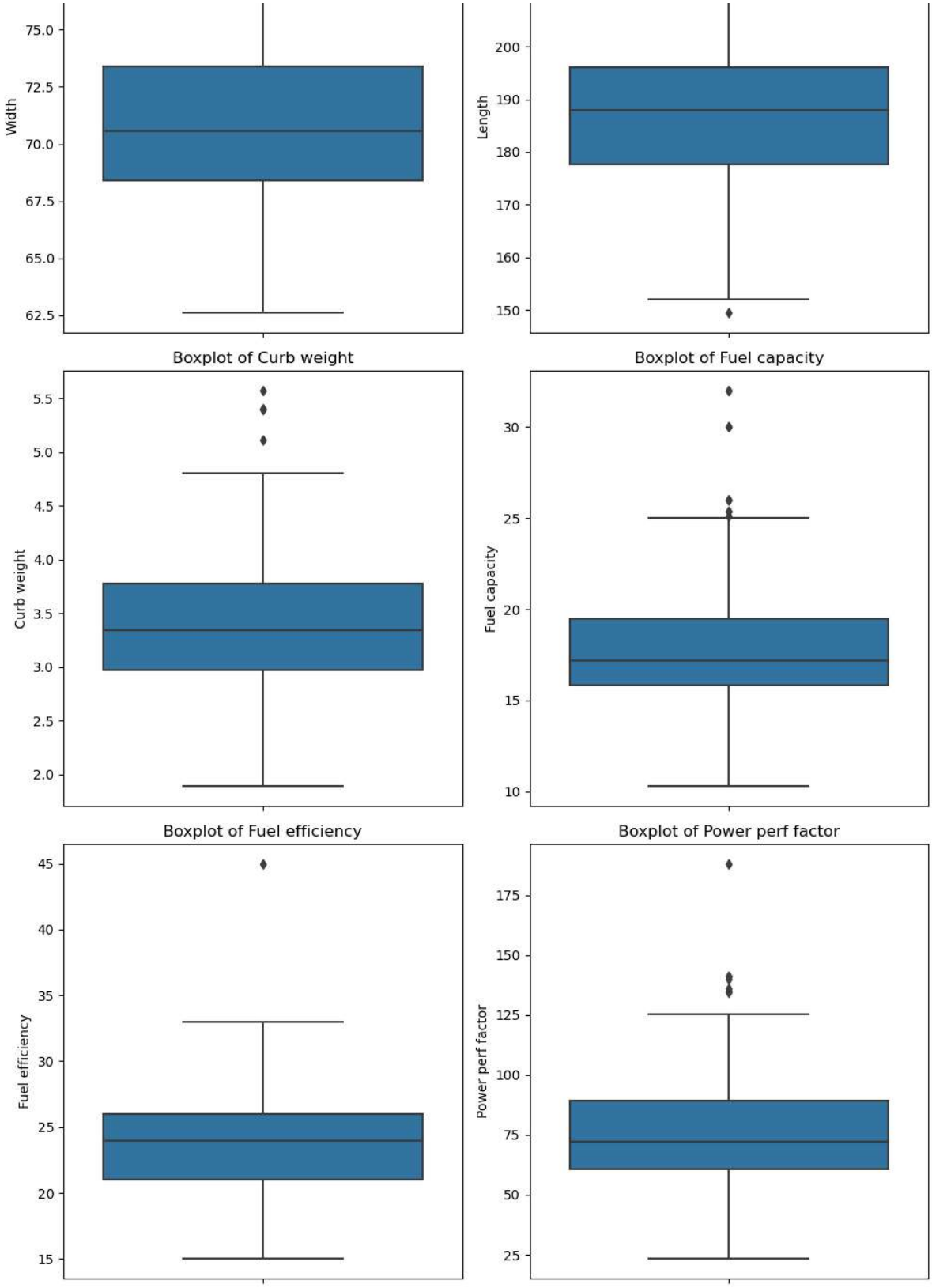
Boxplot of Engine size
Engine size
1
2
3
4
5
6
7
8


Boxplot of Horsepower
Horsepower
50
100
150
200
250
300
350
400
450

Boxplot of Wheelbase
Wheelbase
100
110
120
130
140


Boxplot of Width
80.0
77.5

Boxplot of Length
220
210



Z-score

Detect Outliers using Z-Score. (Set threshold =3)

In [25]:
```
def outliers_detection(data, threshold=3):

    z_scores = data.apply(zscore)
    outliers_mask = abs(z_scores) > threshold
    total_outliers = outliers_mask.any(axis=1).sum()
    print(f"Total outliers detected: {total_outliers} out of {len(data)} e
    return outliers_mask


outliers_mask = outliers_detection(df)
```

```
Total outliers detected: 15 out of 157 entries
```

Remove outliers

In [26]:
```
def outliers_removal(data, outliers_mask):

    cleaned_data = data[~outliers_mask.any(axis=1)]
    print(f"Data after removing outliers has {len(cleaned_data)} entries."
    return cleaned_data

df = outliers_removal(df, outliers_mask)
```

```
Data after removing outliers has 142 entries.
```

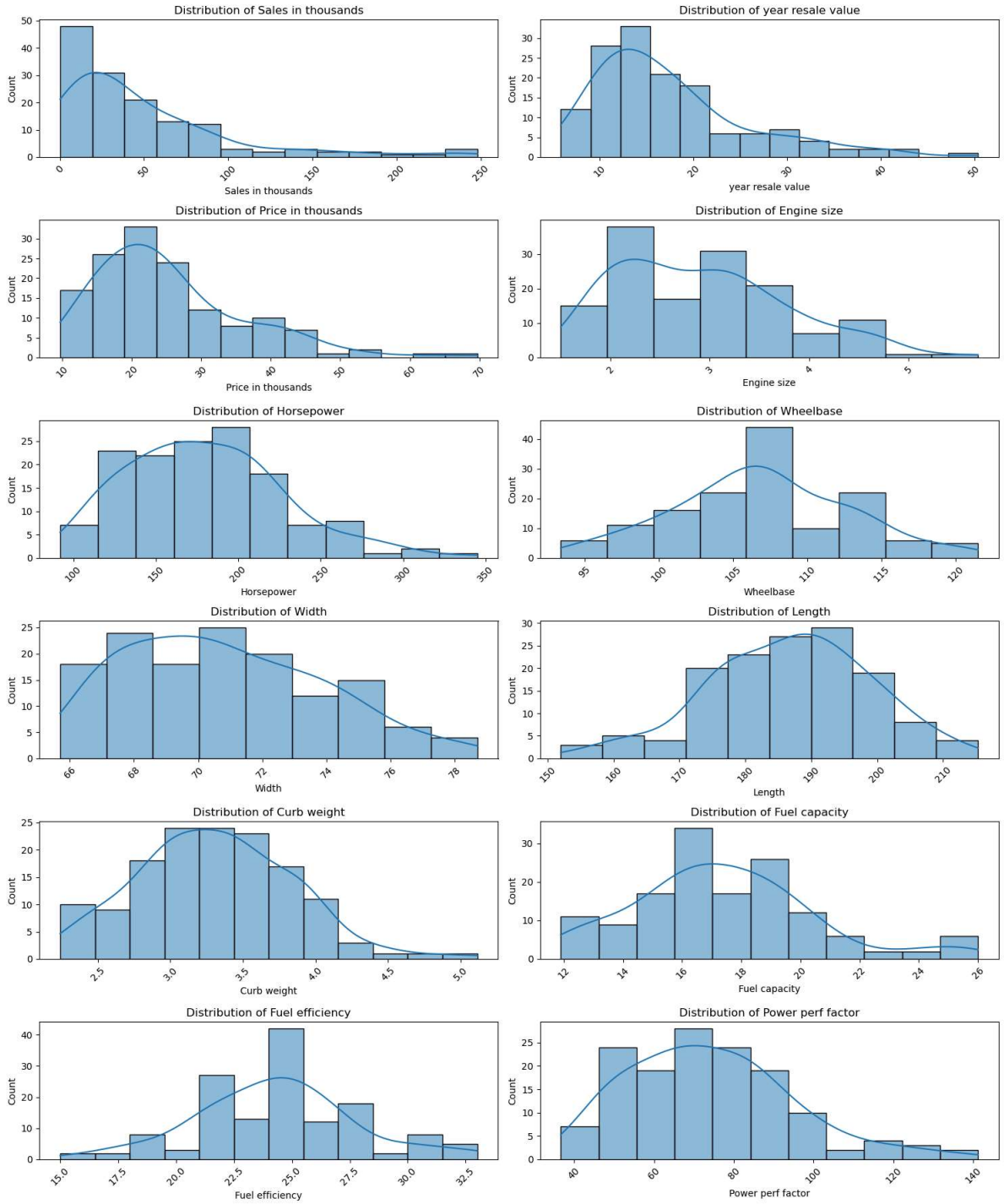
In [27]:
```
def outliers_detection(data, threshold=3):

    z_scores = data.apply(zscore)
    outliers_mask = abs(z_scores) > threshold
    total_outliers = outliers_mask.any(axis=1).sum()
    print(f"Total outliers detected: {total_outliers} out of {len(data)} e
    return outliers_mask


outliers_mask = outliers_detection(df)
```

```
Total outliers detected: 8 out of 142 entries
```

Distribution Check

In [28]:

```
num_columns = len(df.columns)
num_rows = (num_columns + 1) // 2

plt.figure(figsize=(15, 3 * num_rows))
for i, column in enumerate(df.columns):
    plt.subplot(num_rows, 2, i + 1)
    sns.histplot(df[column], kde=True)
    plt.title(f'Distribution of {column}')
    plt.xticks(rotation=45)

plt.tight_layout()
plt.show()
```



Use binning technique to remove Skewness

In [29]:
```
# Apply binning to Sales in thousands, Year resale value, Fuel efficiency
df['Sales in thousands Binned'] = pd.qcut(df['Sales in thousands'], q=100,
df['year resale value Binned'] = pd.qcut(df['year resale value'], q=50, la
df['Price in thousands Binned'] = pd.qcut(df['Price in thousands'], q=100,
df['Fuel efficiency Binned'] = pd.qcut(df['Fuel efficiency'], q=50, labels
df['Power perf factor Binned'] = pd.qcut(df['Power perf factor'], q=30, la
```

In [30]:
```
1 df.head()
```

Out[30]:

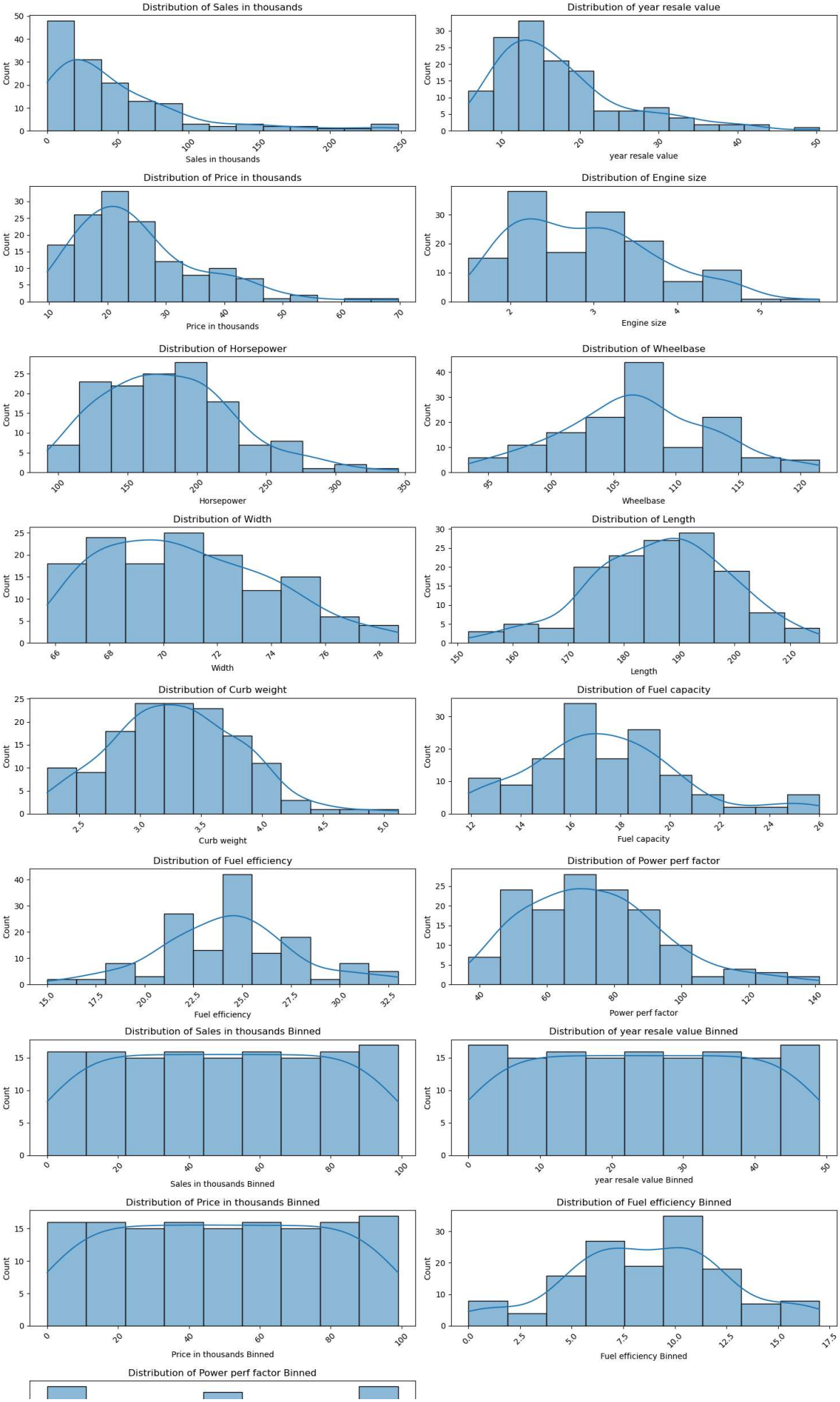
	Sales in thousands	year resale value	Price in thousands	Engine size	Horsepower	Wheelbase	Width	Length	Curb weight	I capa
0	16.919	16.360	21.500	1.8	140.0	101.2	67.3	172.4	2.639	
1	39.384	19.875	28.400	3.2	225.0	108.1	70.3	192.9	3.517	
2	14.114	18.225	22.799	3.2	225.0	106.9	70.6	192.0	3.470	
3	8.588	29.725	42.000	3.5	210.0	114.6	71.4	196.6	3.850	
4	20.397	22.255	23.990	1.8	150.0	102.6	68.2	178.0	2.998	

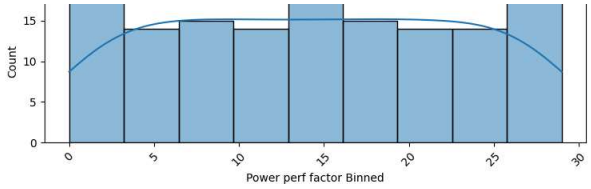
Distribution check

To confirm if skewness is removed or not

In [31]:

```
num_columns = len(df.columns)
num_rows = (num_columns + 1) // 2

plt.figure(figsize=(15, 3 * num_rows))
for i, column in enumerate(df.columns):
    plt.subplot(num_rows, 2, i + 1)
    sns.histplot(df[column], kde=True)
    plt.title(f'Distribution of {column}')
    plt.xticks(rotation=45)

plt.tight_layout()
plt.show()
```


Distribution of Sales in thousands
Sales in thousands
Distribution of year resale value
year resale value
Distribution of Price in thousands
Price in thousands
Distribution of Engine size
Engine size
Distribution of Horsepower
Horsepower
Distribution of Wheelbase
Wheelbase
Distribution of Width
Width
Distribution of Length
Length
Distribution of Curb weight
Curb weight
Distribution of Fuel capacity
Fuel capacity
Distribution of Fuel efficiency
Fuel efficiency
Distribution of Power perf factor
Power perf factor
Distribution of Sales in thousands Binned
Sales in thousands Binned
Distribution of year resale value Binned
year resale value Binned
Distribution of Price in thousands Binned
Price in thousands Binned
Distribution of Fuel efficiency Binned
Fuel efficiency Binned
Distribution of Power perf factor Binned
Count



Drop the skewed columns

In [32]:
```
df.drop(['Sales in thousands', 'year resale value', 'Fuel efficiency',  'F
```

In [33]:
```
df.columns
```

Out[33]:
```
Index(['Engine size', 'Horsepower', 'Wheelbase', 'Width', 'Length',
       'Curb weight', 'Fuel capacity', 'Sales in thousands Binned',
       'year resale value Binned', 'Price in thousands Binned',
       'Fuel efficiency Binned', 'Power perf factor Binned'],
      dtype='object')
```

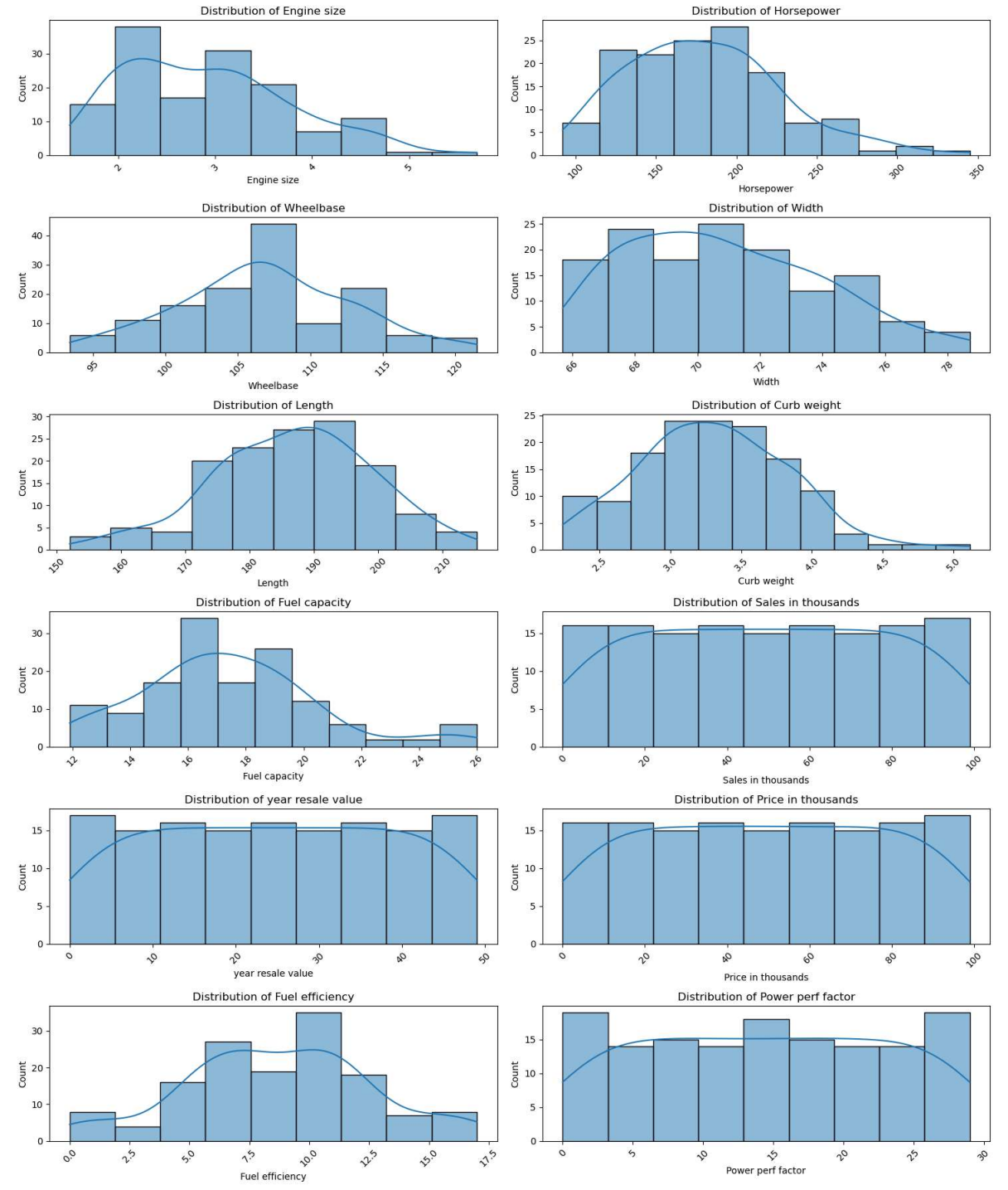
In [34]:
```
df.rename(columns={"Sales in thousands Binned": "Sales in thousands",
                   "year resale value Binned": "year resale value",
                  "Fuel efficiency Binned": "Fuel efficiency",
                   "Power perf factor Binned": "Power perf factor",
                  "Price in thousands Binned": "Price in thousands"},inpla
```

In [35]:
```
df.columns
```

Out[35]:
```
Index(['Engine size', 'Horsepower', 'Wheelbase', 'Width', 'Length',
       'Curb weight', 'Fuel capacity', 'Sales in thousands',
       'year resale value', 'Price in thousands', 'Fuel efficiency',
       'Power perf factor'],
      dtype='object')
```

In [36]:

```
num_columns = len(df.columns)
num_rows = (num_columns + 1) // 2

plt.figure(figsize=(15, 3 * num_rows))
for i, column in enumerate(df.columns):
    plt.subplot(num_rows, 2, i + 1)
    sns.histplot(df[column], kde=True)
    plt.title(f'Distribution of {column}')
    plt.xticks(rotation=45)

plt.tight_layout()
plt.show()
```



Standardization and Normalization

Apply standardization and normalization on the columns

In [37]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler, MinMaxScaler
```

In [38]:
```
df.head()
```

Out[38]:

	Engine size	Horsepower	Wheelbase	Width	Length	Curb weight	Fuel capacity	Sales in thousands	year resale value	Pric thousa
0	1.8	140.0	101.2	67.3	172.4	2.639	13.2	28	27	
1	3.2	225.0	108.1	70.3	192.9	3.517	17.2	58	36	
2	3.2	225.0	106.9	70.6	192.0	3.470	17.2	23	32	
3	3.5	210.0	114.6	71.4	196.6	3.850	18.0	12	44	
4	1.8	150.0	102.6	68.2	178.0	2.998	16.4	36	39	

Standardization

In [39]:
```
scaler = StandardScaler()
df[df.columns] = scaler.fit_transform(df[df.columns])
```

In [40]:
```
df[df.columns].describe().loc[['mean', 'std']]
```

Out[40]:

	Engine size	Horsepower	Wheelbase	Width	Length	Curb weight	Fu
mean	-1.250956e-16	-1.376051e-16	1.063312e-15	3.252484e-16	-3.302523e-15	1.113350e-15	-
std	1.003540e+00	1.003540e+00	1.003540e+00	1.003540e+00	1.003540e+00	1.003540e+00	1.0

Normalization

In [41]:
```
Normalization = MinMaxScaler()
df[df.columns] = Normalization.fit_transform(df[df.columns])
```

In [42]:
```
1 df[df.columns].describe()
```

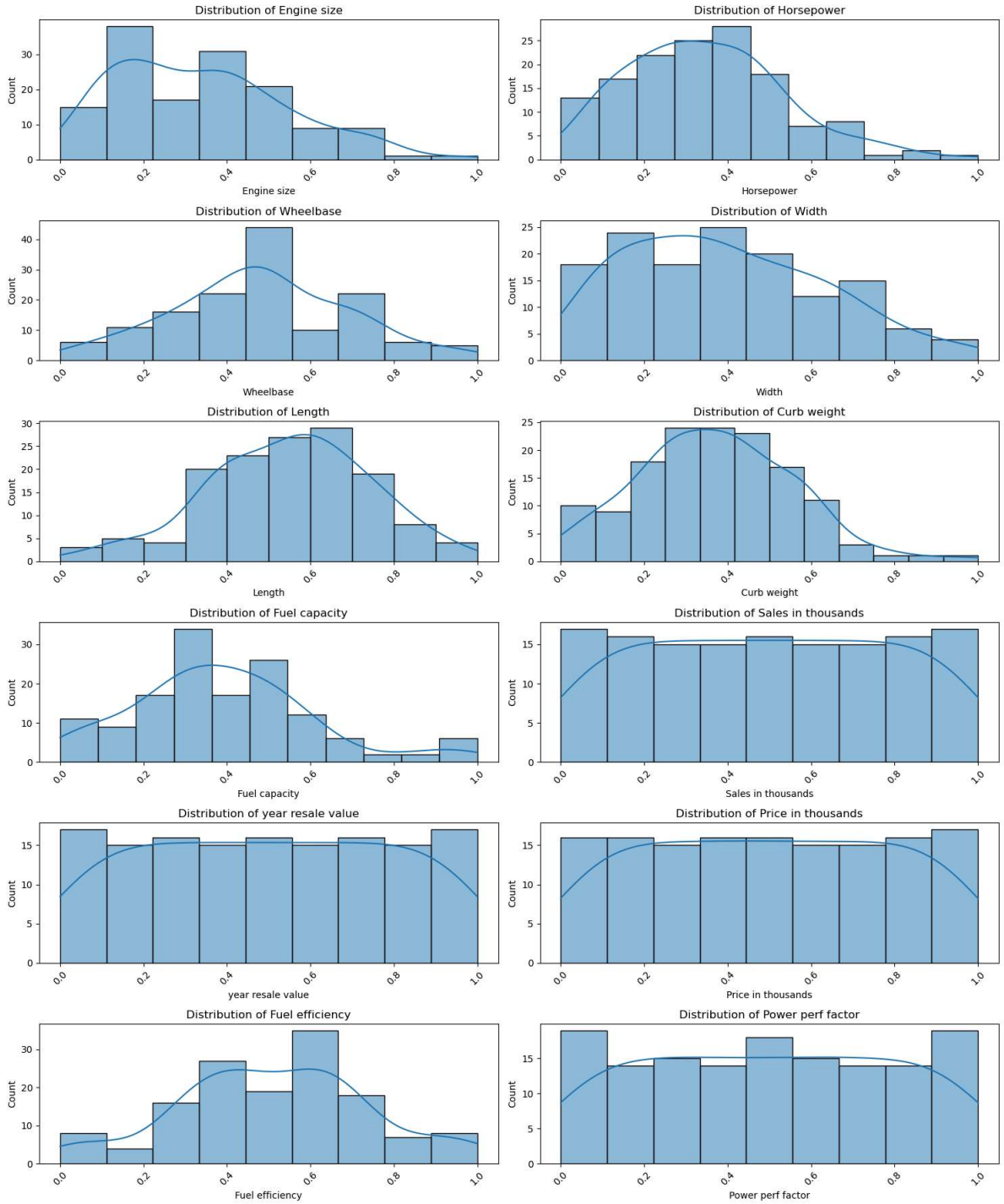
Out[42]:

	Engine size	Horsepower	Wheelbase	Width	Length	Curb weight	Fuel capacity	th
count	142.000000	142.000000	142.000000	142.000000	142.000000	142.000000	142.000000	14
mean	0.338364	0.347868	0.481129	0.394285	0.548105	0.371165	0.387973	
std	0.205774	0.189930	0.211154	0.237420	0.194246	0.185758	0.216553	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.172619	0.215415	0.346085	0.200000	0.410742	0.246348	0.248227	
50%	0.345238	0.324111	0.482206	0.361538	0.547393	0.372870	0.365248	
75%	0.470238	0.466403	0.637011	0.561538	0.675750	0.483043	0.503546	
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

Distribution check

In [43]:

```
num_columns = len(df.columns)
num_rows = (num_columns + 1) // 2

plt.figure(figsize=(15, 3 * num_rows))
for i, column in enumerate(df.columns):
    plt.subplot(num_rows, 2, i + 1)
    sns.histplot(df[column], kde=True)
    plt.title(f'Distribution of {column}')
    plt.xticks(rotation=45)

plt.tight_layout()
plt.show()
```


Distribution of Engine size
Engine size
Distribution of Horsepower
Horsepower
Distribution of Wheelbase
Wheelbase
Distribution of Width
Width
Distribution of Length
Length
Distribution of Curb weight
Curb weight
Distribution of Fuel capacity
Fuel capacity
Distribution of Sales in thousands
Sales in thousands
Distribution of year resale value
year resale value
Distribution of Price in thousands
Price in thousands
Distribution of Fuel efficiency
Fuel efficiency
Distribution of Power perf factor
Power perf factor
Count

The END!