# Exercise - Visualization of Data

## Step 1. Import the necessary libraries

In [4]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(color_codes=True)
```

## Step 2. Import the dataset from the AutoMPG dataset file. Assign it to a variable called mpg_df

In [5]:

```
import pandas as pd

# Read the dataset from the CSV file and assign it to the variable mpg_df
mpg_df = pd.read_csv('Automobile.csv')
```

## Step 3. Perform basic EDA to understand the structure of the data

In [6]:

```
print(mpg_df.head())
```

Loading [MathJax]/extensions/Safe.js

```
   symboling  normalized_losses         make fuel_type aspiration  \
0          3                168  alfa-romero       gas        std   
1          3                168  alfa-romero       gas        std   
2          1                168  alfa-romero       gas        std   
3          2                164         audi       gas        std   
4          2                164         audi       gas        std   

  number_of_doors   body_style drive_wheels engine_location  wheel_base  ...  
\
0             two  convertible          rwd           front        88.6  ...   
1             two  convertible          rwd           front        88.6  ...   
2             two    hatchback          rwd           front        94.5  ...   
3            four        sedan          fwd           front        99.8  ...   
4            four        sedan          4wd           front        99.4  ...   

   engine_size  fuel_system  bore  stroke compression_ratio horsepower  \
0          130         mpfi  3.47    2.68               9.0        111   
1          130         mpfi  3.47    2.68               9.0        111   
2          152         mpfi  2.68    3.47               9.0        154   
3          109         mpfi  3.19    3.40              10.0        102   
4          136         mpfi  3.19    3.40               8.0        115   

   peak_rpm city_mpg  highway_mpg  price  
0      5000       21           27  13495  
1      5000       21           27  16500  
2      5000       19           26  16500  
3      5500       24           30  13950  
4      5500       18           22  17450  

[5 rows x 26 columns]
```

In [7]:

```python
print(mpg_df.describe())
```

```
        symboling  normalized_losses  wheel_base      length       width  \
count  201.000000         201.000000  201.000000  201.000000  201.000000
mean     0.840796         125.189055   98.797015  174.200995   65.889055
std      1.254802          33.572966    6.066366   12.322175    2.101471
min     -2.000000          65.000000   86.600000  141.100000   60.300000
25%      0.000000         101.000000   94.500000  166.800000   64.100000
50%      1.000000         122.000000   97.000000  173.200000   65.500000
75%      2.000000         150.000000  102.400000  183.500000   66.600000
max      3.000000         256.000000  120.900000  208.100000   72.000000

           height  curb_weight  engine_size        bore      stroke  \
count  201.000000   201.000000   201.000000  201.000000  201.000000
mean    53.766667  2555.666667   126.875622    3.329701    3.261741
std      2.447822   517.296727    41.546834    0.268166    0.317875
min     47.800000  1488.000000    61.000000    2.540000    2.070000
25%     52.000000  2169.000000    98.000000    3.150000    3.110000
50%     54.100000  2414.000000   120.000000    3.310000    3.290000
75%     55.500000  2926.000000   141.000000    3.580000    3.460000
max     59.800000  4066.000000   326.000000    3.940000    4.170000

       compression_ratio  horsepower     peak_rpm    city_mpg  highway_mpg  \
count         201.000000  201.000000   201.000000  201.000000   201.000000
mean           10.164279  103.263682  5121.393035   25.179104    30.686567
std             4.004965   37.389372   479.624905    6.423220     6.815150
min             7.000000   48.000000  4150.000000   13.000000    16.000000
25%             8.600000   70.000000  4800.000000   19.000000    25.000000
50%             9.000000   95.000000  5200.000000   24.000000    30.000000
75%             9.400000  116.000000  5500.000000   30.000000    34.000000
max            23.000000  262.000000  6600.000000   49.000000    54.000000

              price
count    201.000000
mean   13207.129353
std     7947.066342
min     5118.000000
25%     7775.000000
50%    10295.000000
75%    16500.000000
max    45400.000000
```

```
In [8]: print(mpg_df.info())
```

Loading [MathJax]/extensions/Safe.js

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201 entries, 0 to 200
Data columns (total 26 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   symboling            201 non-null    int64  
 1   normalized_losses    201 non-null    int64  
 2   make                 201 non-null    object 
 3   fuel_type            201 non-null    object 
 4   aspiration           201 non-null    object 
 5   number_of_doors      201 non-null    object 
 6   body_style           201 non-null    object 
 7   drive_wheels         201 non-null    object 
 8   engine_location      201 non-null    object 
 9   wheel_base           201 non-null    float64
 10  length               201 non-null    float64
 11  width                201 non-null    float64
 12  height               201 non-null    float64
 13  curb_weight          201 non-null    int64  
 14  engine_type          201 non-null    object 
 15  number_of_cylinders  201 non-null    object 
 16  engine_size          201 non-null    int64  
 17  fuel_system          201 non-null    object 
 18  bore                 201 non-null    float64
 19  stroke               201 non-null    float64
 20  compression_ratio    201 non-null    float64
 21  horsepower           201 non-null    int64  
 22  peak_rpm             201 non-null    int64  
 23  city_mpg             201 non-null    int64  
 24  highway_mpg          201 non-null    int64  
 25  price                201 non-null    int64  
dtypes: float64(7), int64(9), object(10)
memory usage: 41.0+ KB
None
```

In [9]:

```python
print(mpg_df.isnull().sum())
```

```
symboling              0
normalized_losses      0
make                   0
fuel_type              0
aspiration             0
number_of_doors        0
body_style             0
drive_wheels           0
engine_location        0
wheel_base             0
length                 0
width                  0
height                 0
curb_weight            0
engine_type            0
number_of_cylinders    0
engine_size            0
fuel_system            0
bore                   0
stroke                 0
compression_ratio      0
horsepower             0
peak_rpm               0
city_mpg               0
highway_mpg            0
price                  0
dtype: int64
```

In [ ]:

```

```

## Step 4. Check and handle the missing values, if any.

In [10]:

```python
missing_values = mpg_df.isnull().sum()
print(missing_values)
```

```
symboling             0
normalized_losses     0
make                  0
fuel_type             0
aspiration            0
number_of_doors       0
body_style            0
drive_wheels          0
engine_location       0
wheel_base            0
length                0
width                 0
height                0
curb_weight           0
engine_type           0
number_of_cylinders   0
engine_size           0
fuel_system           0
bore                  0
stroke                0
compression_ratio     0
horsepower            0
peak_rpm              0
city_mpg              0
highway_mpg           0
price                 0
dtype: int64
```

In [11]:

```
# Replace missing values in a numerical column with its median
median_mpg = mpg_df['city_mpg'].median()
mpg_df['city_mpg'].fillna(median_mpg, inplace=True)
```

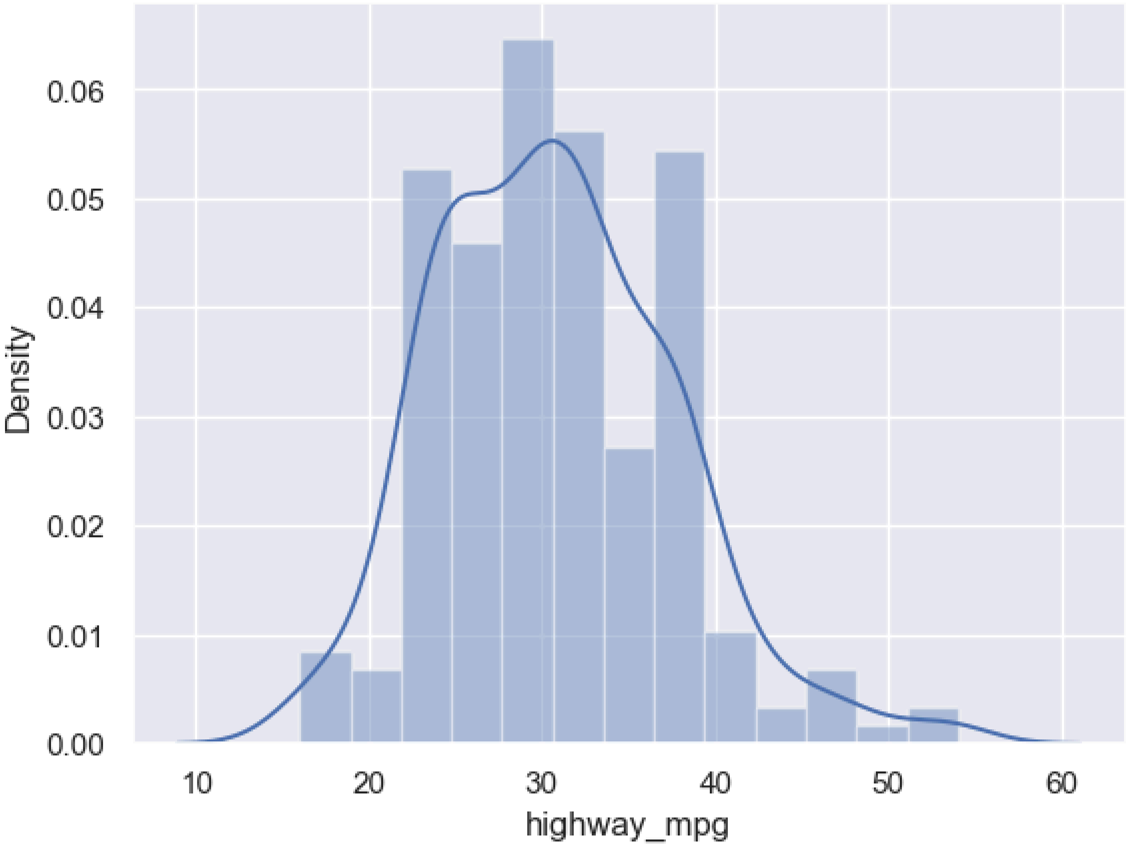
## Step 5. Create a plot to check the relationship between horsepower and acceleration. Note down your insight for the same. Beautify the graph using various customizations.

In [12]:

```
sns.distplot(mpg_df["highway_mpg"])
plt.show()
```

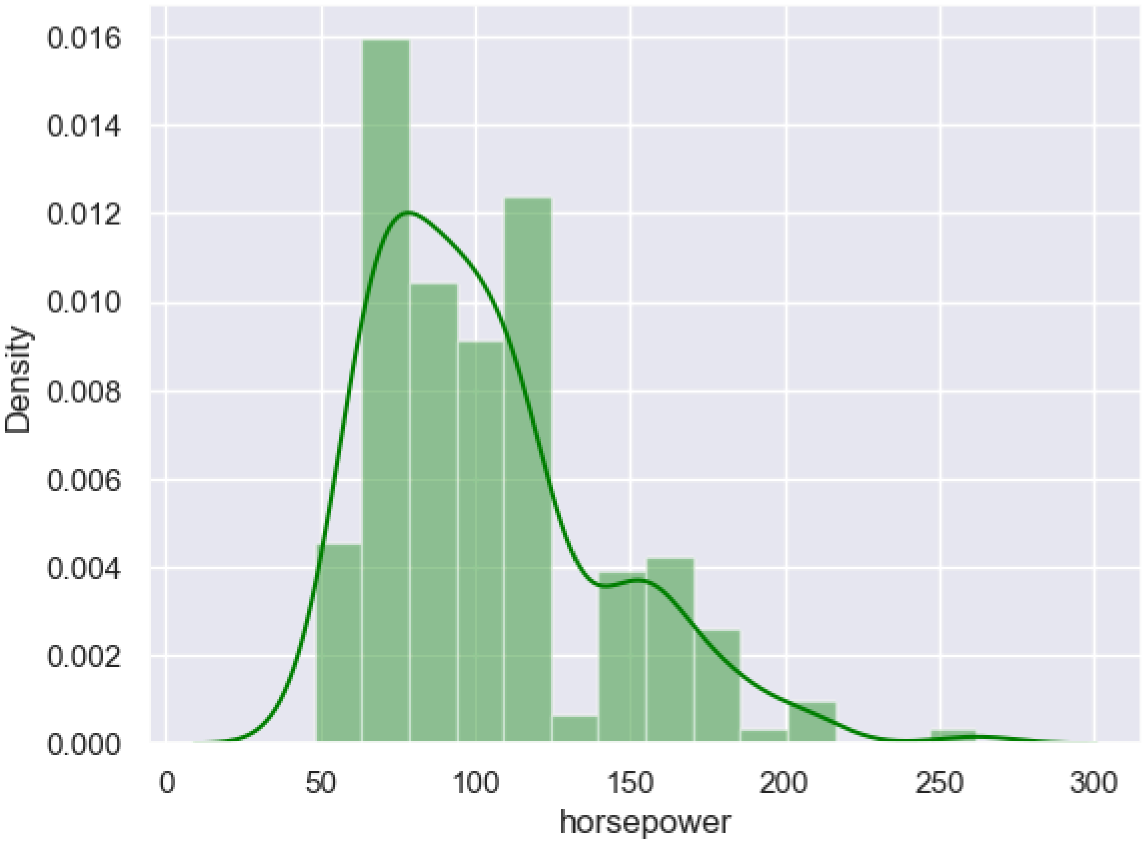
```
C:\Users\intel\AppData\Local\Temp\ipykernel_9832\3846147269.py:1: UserWarnin
g:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(mpg_df["highway_mpg"])
```

Loading [MathJax]/extensions/Safe.js



```
In [13]: sns.distplot(mpg_df['horsepower'],color="green")
         plt.show()
```

```
C:\Users\intel\AppData\Local\Temp\ipykernel_9832\1362431889.py:1: UserWarnin
g:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(mpg_df['horsepower'],color="green")
```



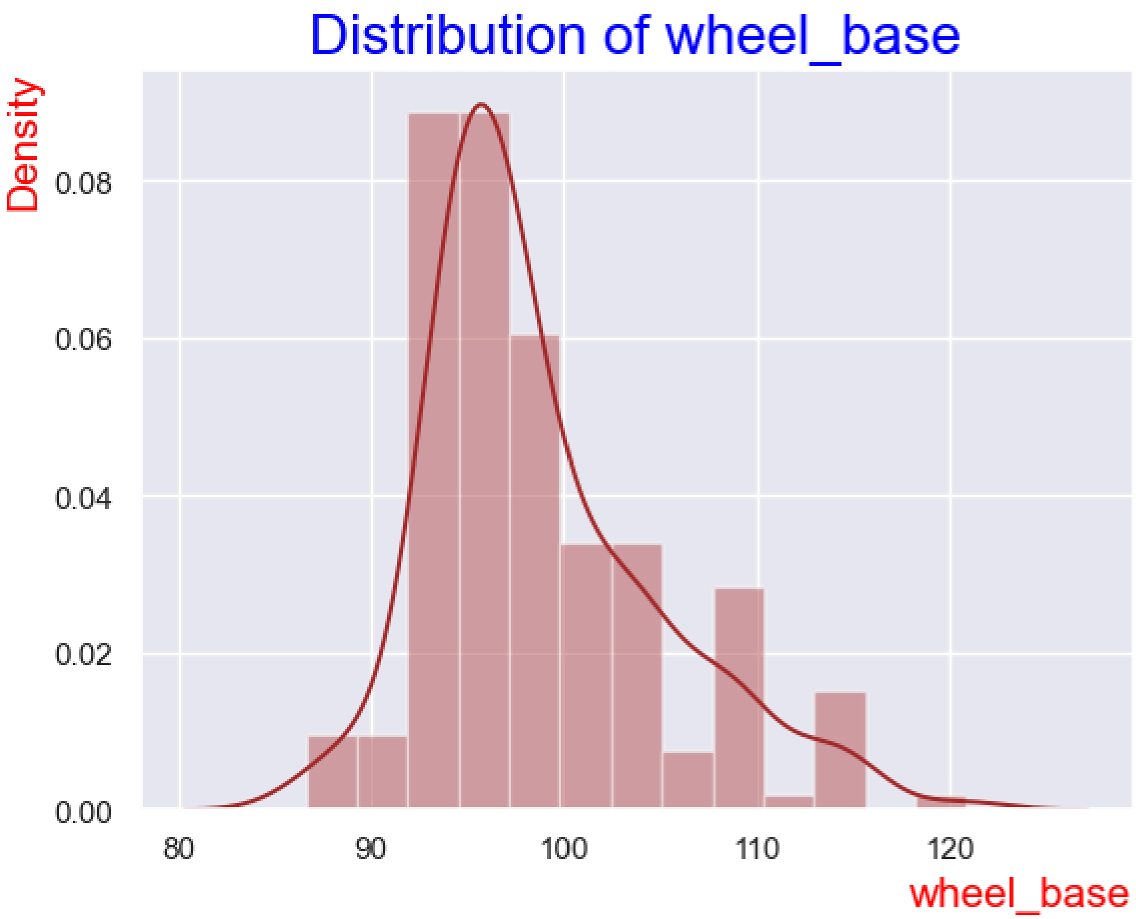
In [ ]:

### Step 6. Generate subplots to display the histograms for acceleration, displacement and weight and kilometer_per_litre. Note down your insight for the same

In [14]:

```python
sns.distplot(mpg_df['wheel_base'],color="brown")
plt.title("Distribution of wheel_base",color="blue",fontsize=20)
plt.xlabel("wheel_base",color="red",fontsize=15,loc="right")
plt.ylabel("Density",color="red",fontsize=15,loc="top")
plt.show()
```

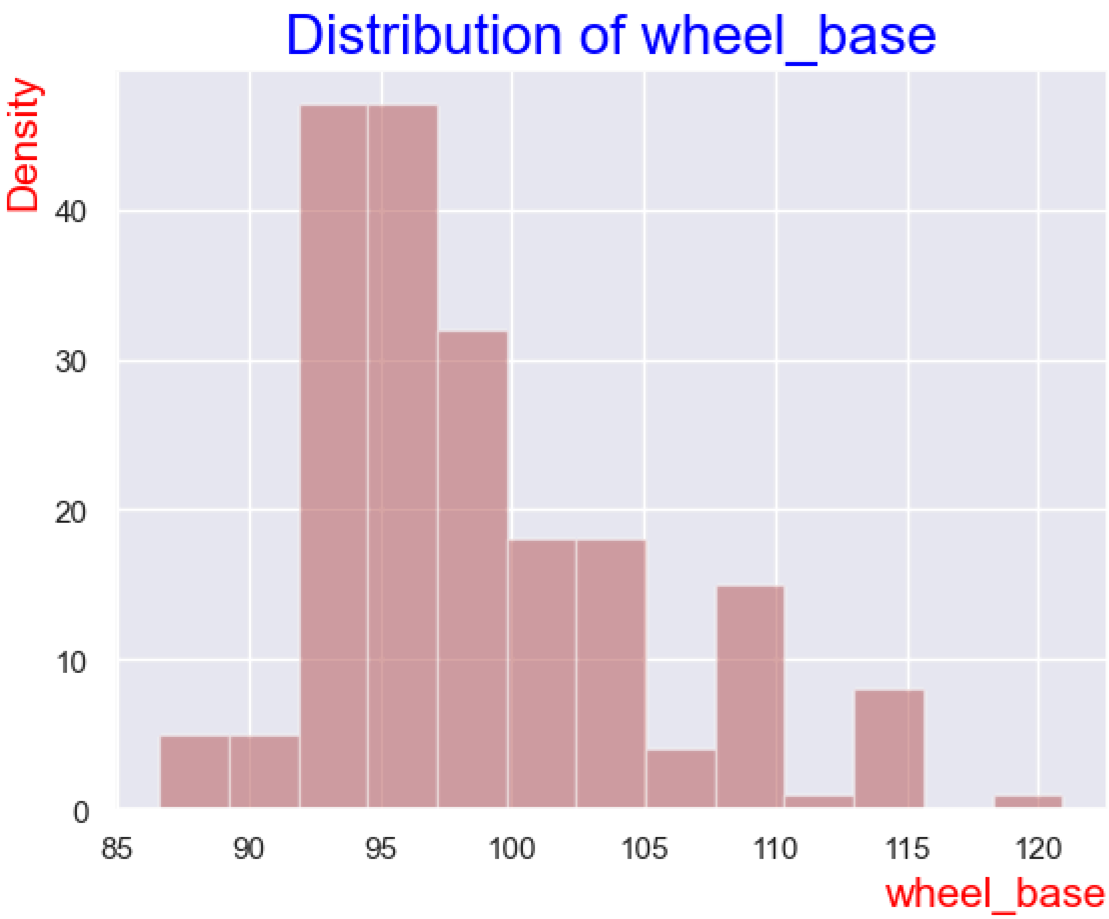
```
C:\Users\intel\AppData\Local\Temp\ipykernel_9832\1642653563.py:1: UserWarnin
g:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(mpg_df['wheel_base'],color="brown")
```



In [ ]:

In [15]:
```
sns.distplot(mpg_df['wheel_base'],kde=False,color="brown")
plt.title("Distribution of wheel_base",color="blue",fontsize=20)
plt.xlabel("wheel_base",color="red",fontsize=15,loc="right")
plt.ylabel("Density",color="red",fontsize=15,loc="top")
plt.show()
```

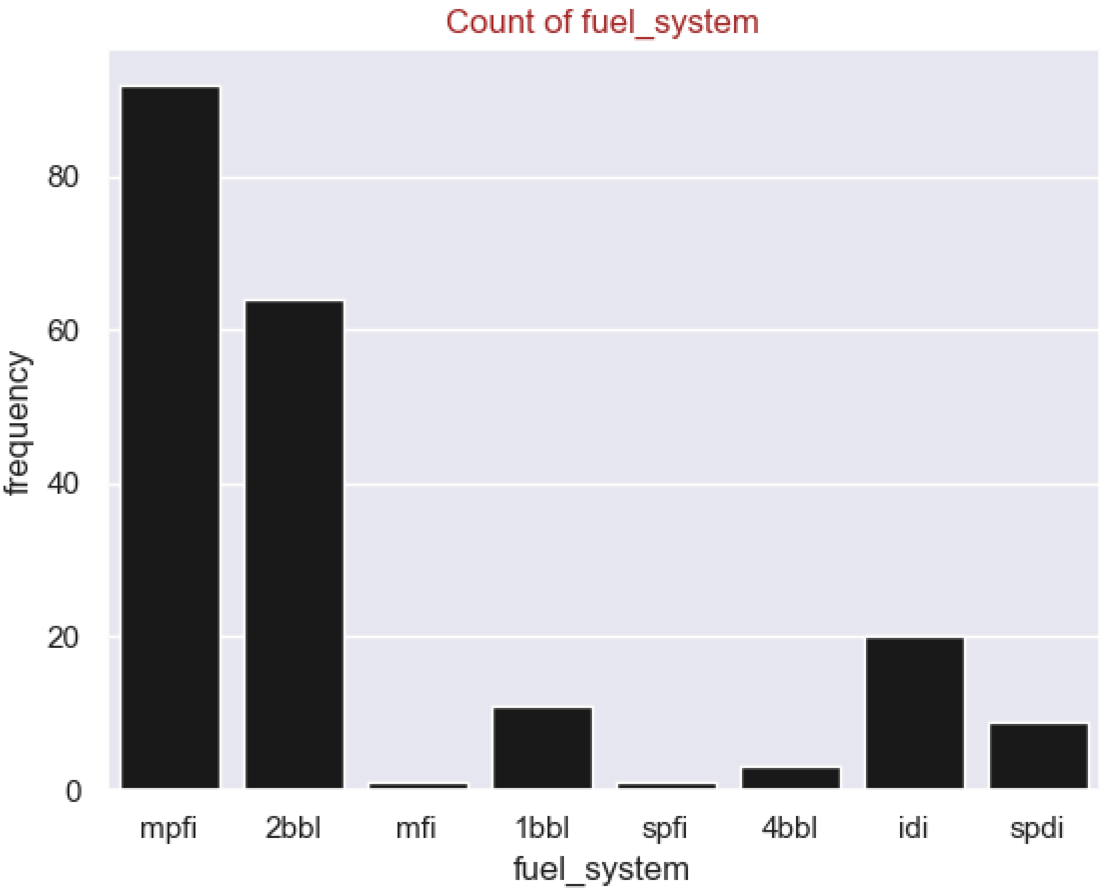
```
C:\Users\intel\AppData\Local\Temp\ipykernel_9832\672822575.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(mpg_df['wheel_base'],kde=False,color="brown")
```



Step 7. Create a plot to check the relationship between displacement and weight for origin 2, color the datapoints based on no of cylinders. Note down your insight for the same and save the plot as an image file.

In [16]:

```
sns.countplot(x="fuel_system",data=mpg_df,color="k")
plt.title("Count of fuel_system",color="brown")
plt.xlabel("fuel_system")
plt.ylabel("frequency")
plt.show()
```



In [17]:

```
mpg_df["fuel_system"].value_counts()
```

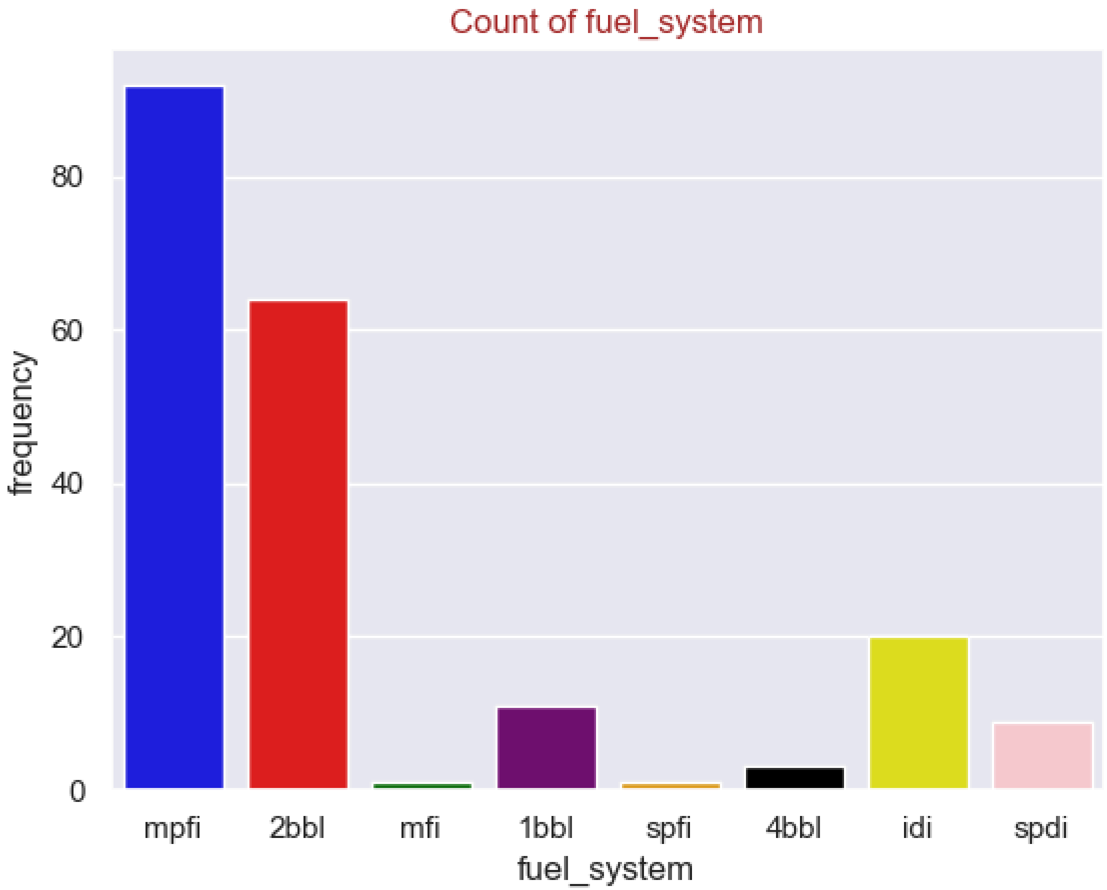
Out[17]:

```
mpfi    92
2bbl    64
idi     20
1bbl    11
spdi     9
4bbl     3
mfi      1
spfi     1
Name: fuel_system, dtype: int64
```

## Step 8. Display the frequency distribution of the Origin variable. Note down your insight for the same

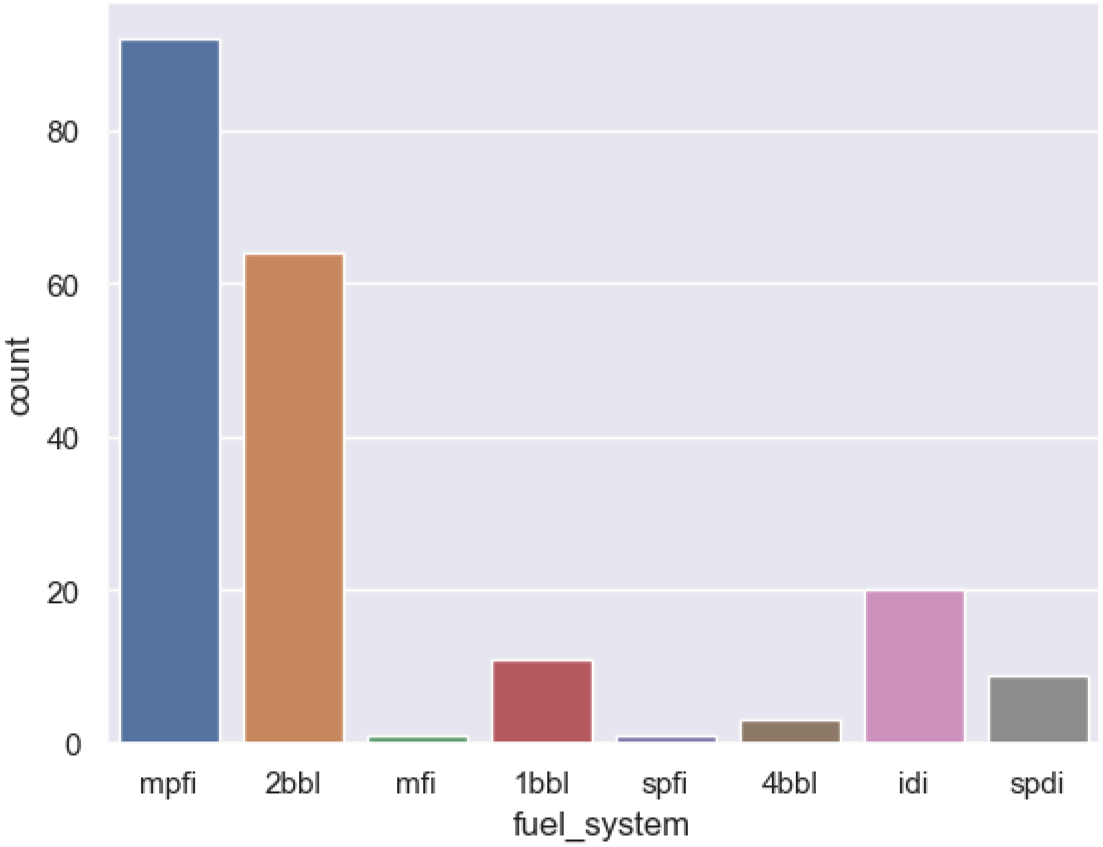
In [18]:

```
sns.countplot(x="fuel_system",data=mpg_df,palette=["blue","red","green","pur
plt.title("Count of fuel_system",color="brown")
plt.xlabel("fuel_system")
plt.ylabel("frequency")
plt.show()
```



## Step 9. Check the relationship of multiple variables wrt kilometer_per_litre. Note down your insight for the same

In [19]:
```
sns.countplot(x="fuel_system",data=mpg_df)
plt.show()
```

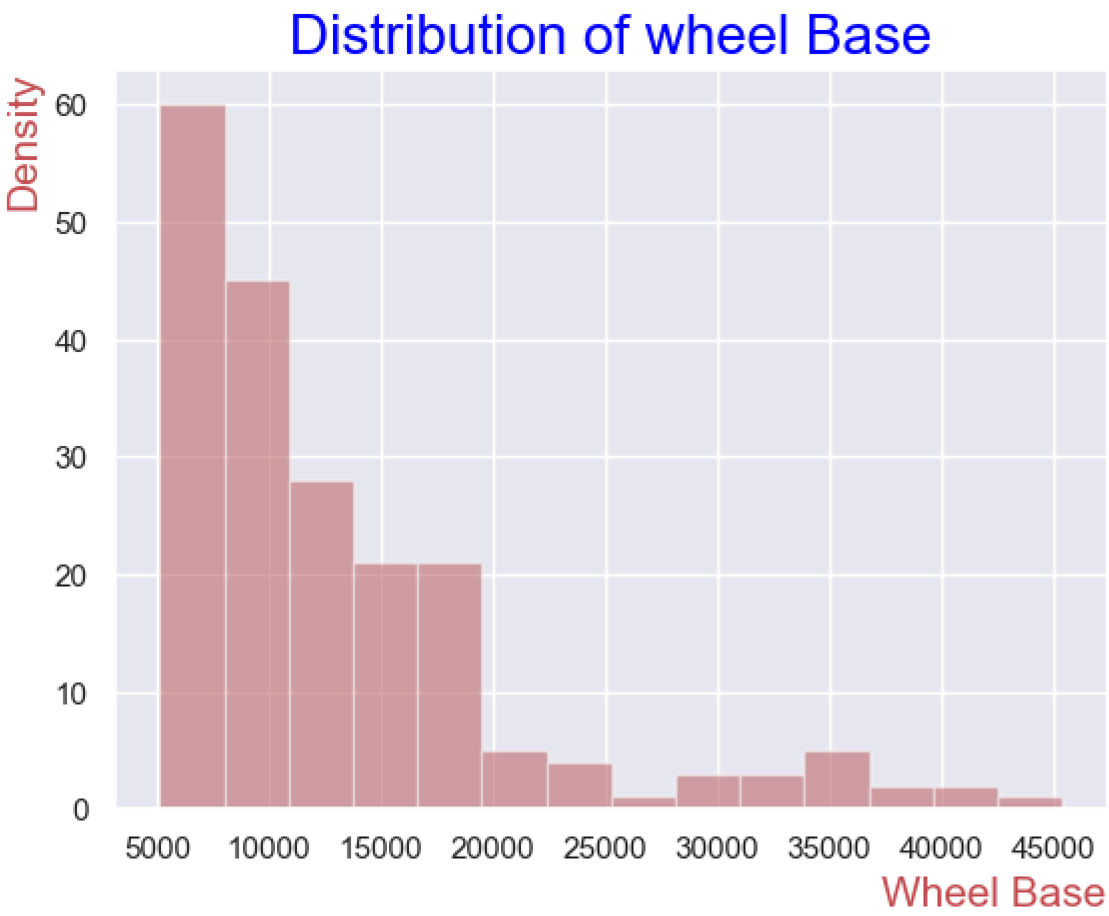


## Step 10. Display the average weight based on no of cylinders present. Note down the insight for the same

In [20]:

```
sns.distplot(mpg_df['price'],kde=False,color="brown")
plt.title(label="Distribution of wheel Base",color="Blue",fontsize=20)
plt.xlabel("Wheel Base",color="r",fontsize=15,loc='right')
plt.ylabel("Density",color="r",fontsize=15,loc='top')

plt.show()
```

```
C:\Users\intel\AppData\Local\Temp\ipykernel_9832\2582493940.py:1: UserWarnin
g:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(mpg_df['price'],kde=False,color="brown")
```



### Step 11. Check for the outliers in the dataset. Note down the insight for the same.

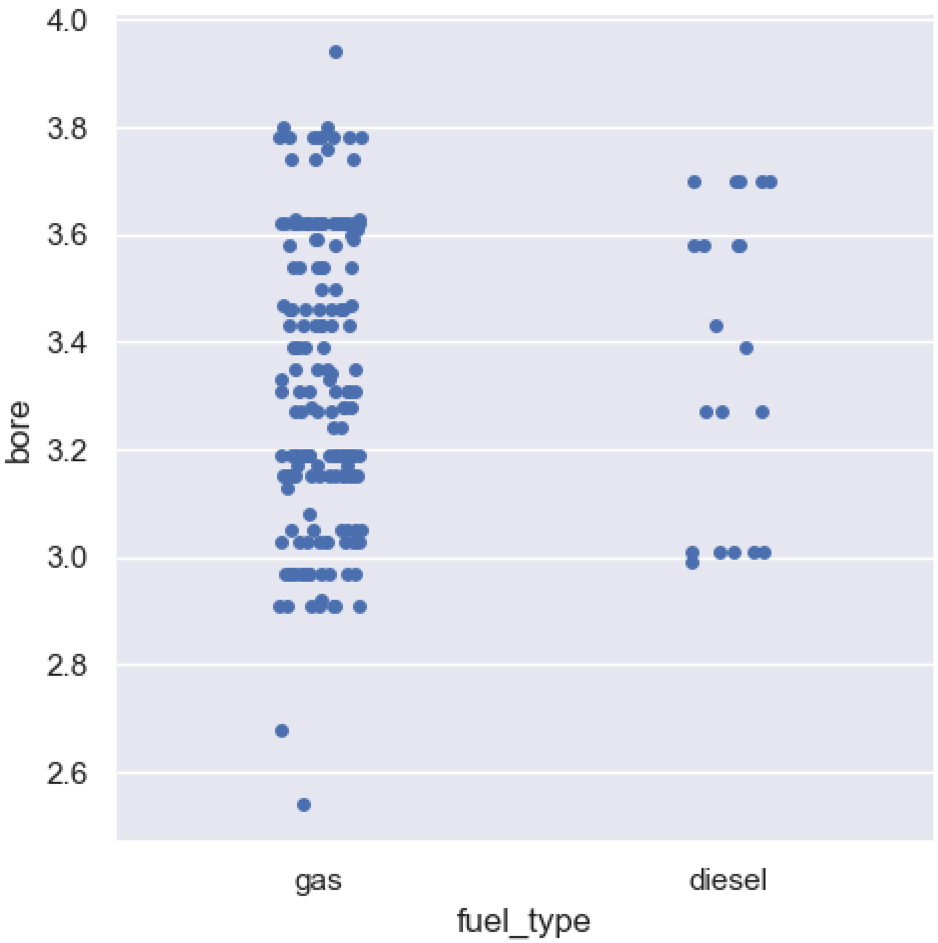
In [ ]:

```
sns.catplot(x="fuel_type",y="bore",kind="swarm",data=mpg_df,palette="Set2")
plt.show()
```

### Step 12. Plot the correlations for variables.

In [29]:

```
sns.catplot(x="fuel_type",y="bore",data=mpg_df)
plt.show()
```



```
In [ ]:
```

```
In [ ]:
```