Data Analysis for Automobile car dealership with Python

1. Exploring the data for Automobile car dealership

In [2]:
```python
# Import pandas library
import  pandas as pd
```

In [3]:
```python
#The online  file by the URL provides  above ,and  assign  it to variable "df"
other_path = "https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/DA0101EN/auto.csv"
df = pd.read_csv(other_path, header=None)
```

In [4]:
```python
# Showing the summary of the  first 5 row using dataframe.head()
print("The first 5 rows of the dataframe")
df.head(5)
```

```
The first 5 rows of the dataframe
```

Out[4]:

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21	22	23	24	25
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500	24	30	13950
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500	18	22	17450

5 rows × 26 columns

In [5]:
```python
#Showing the bottom 10 rows

df.tail(10)
```

Out[5]:

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21	22	23	24	25
195	-1	74	volvo	gas	std	four	wagon	rwd	front	104.3	...	141	mpfi	3.78	3.15	9.5	114	5400	23	28	13415
196	-2	103	volvo	gas	std	four	sedan	rwd	front	104.3	...	141	mpfi	3.78	3.15	9.5	114	5400	24	28	15985
197	-1	74	volvo	gas	std	four	wagon	rwd	front	104.3	...	141	mpfi	3.78	3.15	9.5	114	5400	24	28	16515
198	-2	103	volvo	gas	turbo	four	sedan	rwd	front	104.3	...	130	mpfi	3.62	3.15	7.5	162	5100	17	22	18420
199	-1	74	volvo	gas	turbo	four	wagon	rwd	front	104.3	...	130	mpfi	3.62	3.15	7.5	162	5100	17	22	18950
200	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114	5400	23	28	16845
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	8.7	160	5300	19	25	19045
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	173	mpfi	3.58	2.87	8.8	134	5500	18	23	21485
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1	...	145	idi	3.01	3.40	23.0	106	4800	26	27	22470
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114	5400	19	25	22625

10 rows × 26 columns

In [7]:
```python
# creating headers for the rows.

headers = ["symboling","normalized-loses","make","fuel-type","aspiration","num-of-doors","body-style","drive-wheels","engine-location","wheel_base","length","width","height","curb-weight","engine-type",
         "num-of-cylinders","engine-size","fuel-system","bore","stroke","compresion-rate","horsepower","peak-rpm","city-mpg","highway-mpg","price"]

print("headers\n" , headers)
```

```
headers
 ['symboling', 'normalized-loses', 'make', 'fuel-type', 'aspiration', 'num-of-doors', 'body-style', 'drive-wheels', 'engine-location', 'wheel_base', 'length', 'width', 'height', 'curb-weight', 'engine-type', 'num-of-cylind
ers', 'engine-size', 'fuel-system', 'bore', 'stroke', 'compresion-rate', 'horsepower', 'peak-rpm', 'city-mpg', 'highway-mpg', 'price']
```

In [8]:
```python
# I will replace headers and recheck the data frame
df.columns= headers
df.head(10)
```

Out[8]:

	symboling	normalized-loses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel_base	...	engine-size	fuel-system	bore	stroke	compresion-rate	horsepower	peak-rpm	city-mpg	highway-mpg	price
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500	24	30	13950
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500	18	22	17450
5	2	?	audi	gas	std	two	sedan	fwd	front	99.8	...	136	mpfi	3.19	3.40	8.5	110	5500	19	25	15250
6	1	158	audi	gas	std	four	sedan	fwd	front	105.8	...	136	mpfi	3.19	3.40	8.5	110	5500	19	25	17710
7	1	?	audi	gas	std	four	wagon	fwd	front	105.8	...	136	mpfi	3.19	3.40	8.5	110	5500	19	25	18920
8	1	158	audi	gas	turbo	four	sedan	fwd	front	105.8	...	131	mpfi	3.13	3.40	8.3	140	5500	17	20	23875
9	0	?	audi	gas	turbo	two	hatchback	4wd	front	99.5	...	131	mpfi	3.13	3.40	7.0	160	5500	16	22	?

10 rows × 26 columns

In [9]:
```python
# Dropping missing values  along the  column "price" so that ican focus on the cars that were bought.
df.dropna(subset=["price"], axis =0)
```

Out[9]:

	symboling	normalized-loses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel_base	...	engine-size	fuel-system	bore	stroke	compresion-rate	horsepower	peak-rpm	city-mpg	highway-mpg	price
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500	24	30	13950
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500	18	22	17450
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
200	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114	5400	23	28	16845
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	8.7	160	5300	19	25	19045
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	173	mpfi	3.58	2.87	8.8	134	5500	18	23	21485
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1	...	145	idi	3.01	3.40	23.0	106	4800	26	27	22470
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114	5400	19	25	22625

205 rows × 26 columns

In [14]:
```python
# Saving the Data set.
df.to_csv("automobile.csv", index =False)
```

2.Data Wrangling to convert the data from the initial format to a format that may be better for analysis.

In [15]:
```python
# libaries

import pandas  as pd
import matplotlib.pylab as plt
import numpy as np
```

In [16]:
```python
# replace "?" to NaN

df.replace("?",np.nan , inplace =True)
df.head(5)
```

Out[16]:

	symboling	normalized-loses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel_base	...	engine-size	fuel-system	bore	stroke	compresion-rate	horsepower	peak-rpm	city-mpg	highway-mpg	price
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
1	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
2	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500	24	30	13950
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500	18	22	17450

5 rows × 26 columns

In [17]:
```python
# Identifying  missing  values within the data.
missing_data= df.isnull()
missing_data.head(5)
```

Out[17]:

	symboling	normalized-loses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel_base	...	engine-size	fuel-system	bore	stroke	compresion-rate	horsepower	peak-rpm	city-mpg	highway-mpg	price
0	False	True	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
1	False	True	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
2	False	True	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False

5 rows × 26 columns

In [18]:
```python
avg_norm_loss = df["normalized-loses"].astype("float").mean(axis=0)
print("Average of normalized-loses:",avg_norm_loss)
```

```
Average of normalized-loses: 122.0
```

In [19]:
```python
# COUNT MISSING VALUES IN EACH COLUMN
for column in missing_data.columns.values.tolist():
    print(column)
    print(missing_data[column].value_counts())
    print("")
```

```
symboling
False    205
Name: symboling, dtype: int64

normalized-loses
False    164
True      41
Name: normalized-loses, dtype: int64

make
False    205
Name: make, dtype: int64

fuel-type
False    205
Name: fuel-type, dtype: int64

aspiration
False    205
Name: aspiration, dtype: int64

num-of-doors
False    203
True       2
Name: num-of-doors, dtype: int64

body-style
False    205
Name: body-style, dtype: int64

drive-wheels
False    205
Name: drive-wheels, dtype: int64

engine-location
False    205
Name: engine-location, dtype: int64

wheel_base
False    205
Name: wheel_base, dtype: int64

length
False    205
Name: length, dtype: int64

width
False    205
Name: width, dtype: int64

height
False    205
Name: height, dtype: int64

curb-weight
False    205
Name: curb-weight, dtype: int64

engine-type
False    205
Name: engine-type, dtype: int64

num-of-cylinders
False    205
Name: num-of-cylinders, dtype: int64

engine-size
False    205
Name: engine-size, dtype: int64

fuel-system
False    205
Name: fuel-system, dtype: int64

bore
False    201
True       4
Name: bore, dtype: int64

stroke
False    201
True       4
Name: stroke, dtype: int64

compresion-rate
False    205
Name: compresion-rate, dtype: int64

horsepower
False    203
True       2
Name: horsepower, dtype: int64

peak-rpm
False    203
True       2
Name: peak-rpm, dtype: int64

city-mpg
False    205
Name: city-mpg, dtype: int64

highway-mpg
False    205
Name: highway-mpg, dtype: int64

price
False    201
True       4
Name: price, dtype: int64
```

3.Replacing missing values/null values on the data frame

In [20]:
```python
# Calculating average for normalized losses
avg_norm_loss = df['normalized-loses'].astype('float').mean(axis=0)
print("Average of normalized-loses: ",avg_norm_loss)
```

```
Average of normalized-loses:  122.0
```

In [21]:
```python
# Replacing missing value with  normalized losses average on the data frame.
df["normalized-loses"].replace(np.nan,avg_norm_loss, inplace=True)
```

In [22]:
```python
#calculating  average for bore
avg_bore=df['bore'].astype('float').mean(axis=0)
print("Average of bore: ", avg_bore)
```

```
Average of bore:  3.3297512437810943
```

In [23]:
```python
#replacing missing value bore avaerage on the  dataframe
df["bore"].replace(np.nan, avg_bore, inplace=True)
```

In [24]:
```python
#calculating average for stroke
avg_stroke= df['stroke'].astype('float').mean(axis=0)
print('stroke: ', avg_stroke)
```

```
stroke:  3.255422885572139
```

In [25]:
```python
# Replacing missing value with stroke on the dataframe
df["stroke"].replace(np.nan,avg_stroke, inplace=True)
```

In [26]:
```python
# Calculating average  for the  horse power
avg_horsepower = df['horsepower'].astype('float').mean(axis=0)
print("Average of horsepower: ",avg_horsepower)
```

```
Average of horsepower:  104.25615763546799
```

In [27]:
```python
# Replacing missing  value with horsepower  on the dataframe.
df["horsepower"].replace(np.nan, avg_horsepower, inplace=True)
```

In [28]:
```python
# Calculating  the avaerage for peak-rpm
avg_peakrpm =df['peak-rpm'].astype('float').mean(axis=0)
print("peak-rpm: ", avg_peakrpm)
```

```
peak-rpm:  5125.369458128079
```

In [29]:
```python
#Replacing  missing value peak-rpm on the dataframe
df["peak-rpm"].replace(np.nan ,avg_peakrpm, inplace=True)
```

In [30]:
```python
# To see which values are oresentin a particular column

df["num-of-doors"].value_counts()
```

Out[30]:
```
four    114
two      89
Name: num-of-doors, dtype: int64
```

In [31]:
```python
# I can use  the ".idxmax()" method to calculate  the most commontype automatically:
df["num-of-doors"].value_counts().idxmax()
```

Out[31]:
```
'four'
```

In [32]:
```python
# Dropping   the whole column with the missing data.
df.dropna(subset=["price"],axis=0, inplace=True)

#reset index ,because we dropped two rows
df.reset_index(drop=True, inplace=True)
```

In [33]:
```python
df.head()
```

Out[33]:

	symboling	normalized-loses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel_base	...	engine-size	fuel-system	bore	stroke	compresion-rate	horsepower	peak-rpm	city-mpg	highway-mpg	price
0	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
1	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
2	1	122.0	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500	24	30	13950
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500	18	22	17450

5 rows × 26 columns

1. Converting data types to proper formats

In [35]:
```python
# Converting  data
df[["bore","stroke"]] = df[["bore","stroke"]].astype("float")
df[["normalized-loses"]] = df[["normalized-loses"]].astype("int")
df[["price"]] = df[["price"]].astype("float")
df[["peak-rpm"]] =  df[["peak-rpm"]].astype("float")
```

In [37]:
```python
#  list  the columns  after the conversion
df.dtypes
```

Out[37]:
```
symboling             int64
normalized-loses      int32
make                 object
fuel-type            object
aspiration           object
num-of-doors         object
body-style           object
drive-wheels         object
engine-location      object
wheel_base          float64
length              float64
width               float64
height              float64
curb-weight           int64
engine-type          object
num-of-cylinders     object
engine-size           int64
fuel-system          object
bore                float64
stroke              float64
compresion-rate     float64
horsepower           object
peak-rpm            float64
city-mpg              int64
highway-mpg           int64
price               float64
dtype: object
```

1. Example

Transform miles per gallon to Kilometers.

In our dataset,the fuel consumption columns"city-mpg" and "highway-mpg" are represented by mpg(miles per gallon)unit.Assume we are developing an application in a country that accept the fuel consumption with l/100km standard.

We will need to apply data transformation to transform mpg into L/100km?

The formula for unit conversion is

L/100km =235/mpg

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

Out[38]:

	symboling	normalized-loses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel_base	...	engine-size	fuel-system	bore	stroke	compresion-rate	horsepower	peak-rpm	city-mpg	highway-mpg	price
0	3	122	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000.0	21	27	13495.0
1	3	122	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000.0	21	27	16500.0
2	1	122	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000.0	19	26	16500.0
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500.0	24	30	13950.0
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500.0	18	22	17450.0

5 rows × 26 columns

In [39]:
```python
# Convert mpg to L/100km by mathematical operation (235 divided by mpg)
df['city-L/100km'] = 235/df["city-mpg"]

# check tranformed data
df.head()
```

Out[39]:

	symboling	normalized-loses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel_base	...	fuel-system	bore	stroke	compresion-rate	horsepower	peak-rpm	city-mpg	highway-mpg	price	city-L/100km
0	3	122	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	mpfi	3.47	2.68	9.0	111	5000.0	21	27	13495.0	11.190476
1	3	122	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	mpfi	3.47	2.68	9.0	111	5000.0	21	27	16500.0	11.190476
2	1	122	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	mpfi	2.68	3.47	9.0	154	5000.0	19	26	16500.0	12.368421
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	mpfi	3.19	3.40	10.0	102	5500.0	24	30	13950.0	9.791667
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	mpfi	3.19	3.40	8.0	115	5500.0	18	22	17450.0	13.055556

5 rows × 27 columns

In [40]:
```python
# Tranforming mpg to L/100km in the column of " highway-mpg "
df['highway-L/100km'] = 235/df["highway-mpg"]

# Checking tranformed data
df.head()
```

Out[40]:

	symboling	normalized-loses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel_base	...	bore	stroke	compresion-rate	horsepower	peak-rpm	city-mpg	highway-mpg	price	city-L/100km	highway-L/100km
0	3	122	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	3.47	2.68	9.0	111	5000.0	21	27	13495.0	11.190476	8.703704
1	3	122	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	3.47	2.68	9.0	111	5000.0	21	27	16500.0	11.190476	8.703704
2	1	122	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	2.68	3.47	9.0	154	5000.0	19	26	16500.0	12.368421	9.038462
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	3.19	3.40	10.0	102	5500.0	24	30	13950.0	9.791667	7.833333
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	3.19	3.40	8.0	115	5500.0	18	22	17450.0	13.055556	10.681818

5 rows × 28 columns

1. Example : Objective I want to normalize the variables so thier values ranges from 0 to 1. Approach :replace original value by (original value)/(maxium value)

In [41]:
```python
#replace(original value) by (original value)/(maxium value)
df['lenghth'] = df['length']/df['length'].max()
df['width']  = df['width']/df['width'].max()
```

In [42]:
```python
# Normalizing the column "height"
df['height'] = df['height']/df['height'].max()
```

1. Example In our dataset,"horsepoer" is a real valued variable ranging from 48 to 288,it has 57 unique values.What if we only care about the price difference between cars with high horsepower,medium horsepower , and little (3types)?Can we rearrange them into three "bins" to simplify analysis ?

I can only use the method 'cut' to segment the 'horsepower' column into 3 bins

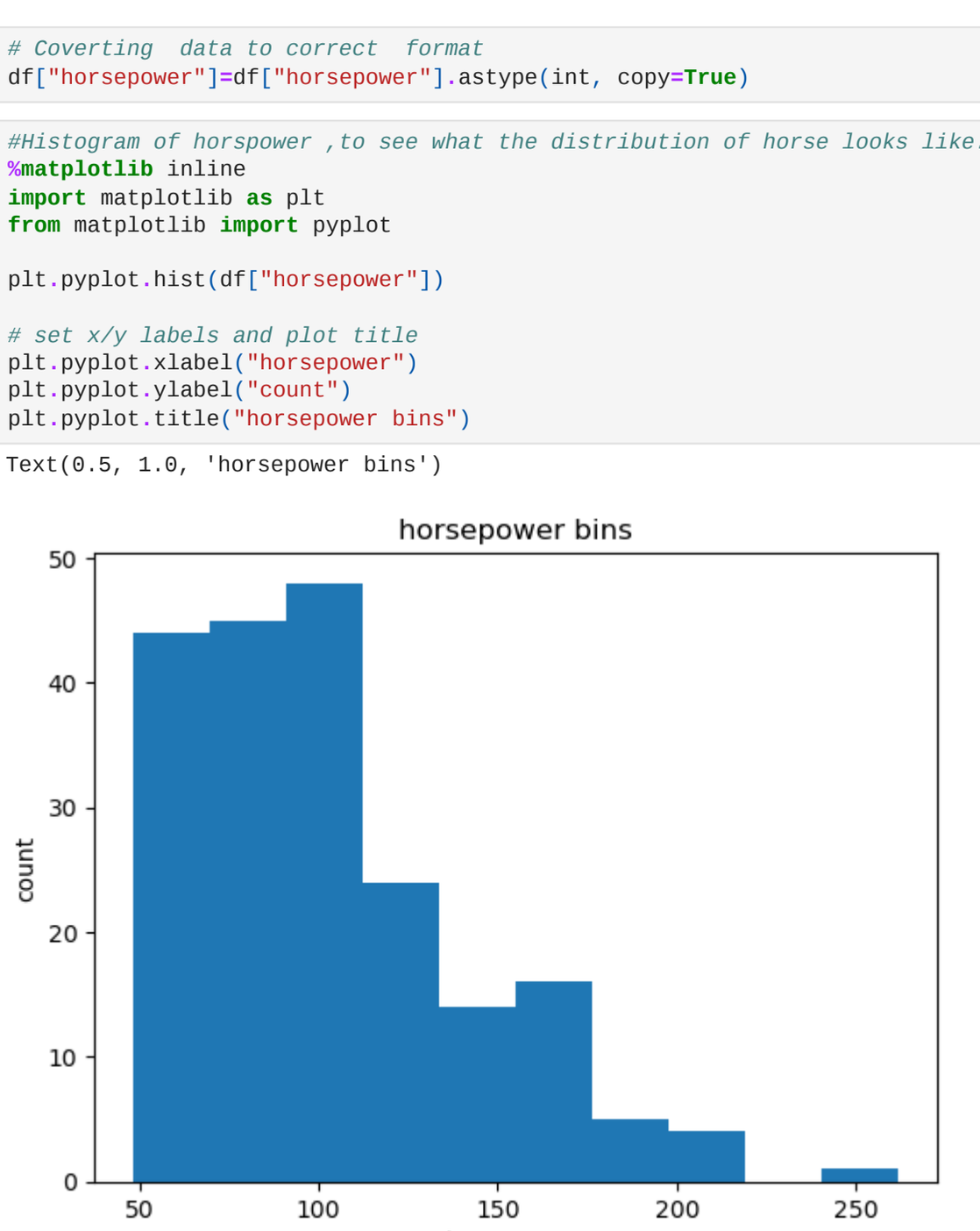
In [44]:
```python
# Coverting  data to correct  format
df["horsepower"]=df["horsepower"].astype(int, copy=True)
```

In [45]:
```python
#Histogram of horspower ,to see what the distribution of horse looks like.
%matplotlib inline
import matplotlib as plt
from matplotlib import pyplot

plt.pyplot.hist(df["horsepower"])

# set x/y labels and plot title
plt.pyplot.xlabel("horsepower")
plt.pyplot.ylabel("count")
plt.pyplot.title("horsepower bins")
```

Out[45]:
```
Text(0.5, 1.0, 'horsepower bins')
```



1. I would like 3 bins of equal size bandwidth so we use numply's linspace(start_valu, end_value, numbers_generated function. Since i want to include the minimum value of horsepower we want to set start_value = min(df["horsepower"]).Also i want to include the maxium value of horsepower we want to set end_value=max(df["horsepower"]).Lastly since am building 3 bins of equal length ,there should be 4 dividers , so numbers _generated =4

In [46]:
```python
# Building array  ,with a minimum value to a maxium value, with bandwith calculated above.The bins will be values used to determine when one bin  ends and another begins.
bins = np.linspace(min(df["horsepower"]),max(df["horsepower"]),4)
bins
```

Out[46]:
```
array([ 48.        , 119.33333333, 190.66666667, 262.        ])
```

In [47]:
```python
# setting group names
group_names = ['Low', 'Medium', 'high']
```

In [48]:
```python
# Applying the function "cut" the determine what each value of "df['horsepower']" belongs to.
df['horsepower-binned']=pd.cut(df['horsepower'],bins, labels=group_names, include_lowest = True )
df[['horsepower','horsepower-binned']].head(20)
```

Out[48]:

	horsepower	horsepower-binned
0	111	Low
1	111	Low
2	154	Medium
3	102	Low
4	115	Low
5	110	Low
6	110	Low
7	110	Low
8	140	Medium
9	101	Low
10	101	Low
11	121	Medium
12	121	Medium
13	121	Medium
14	182	Medium
15	182	Medium
16	182	Medium
17	48	Low
18	70	Low
19	70	Low

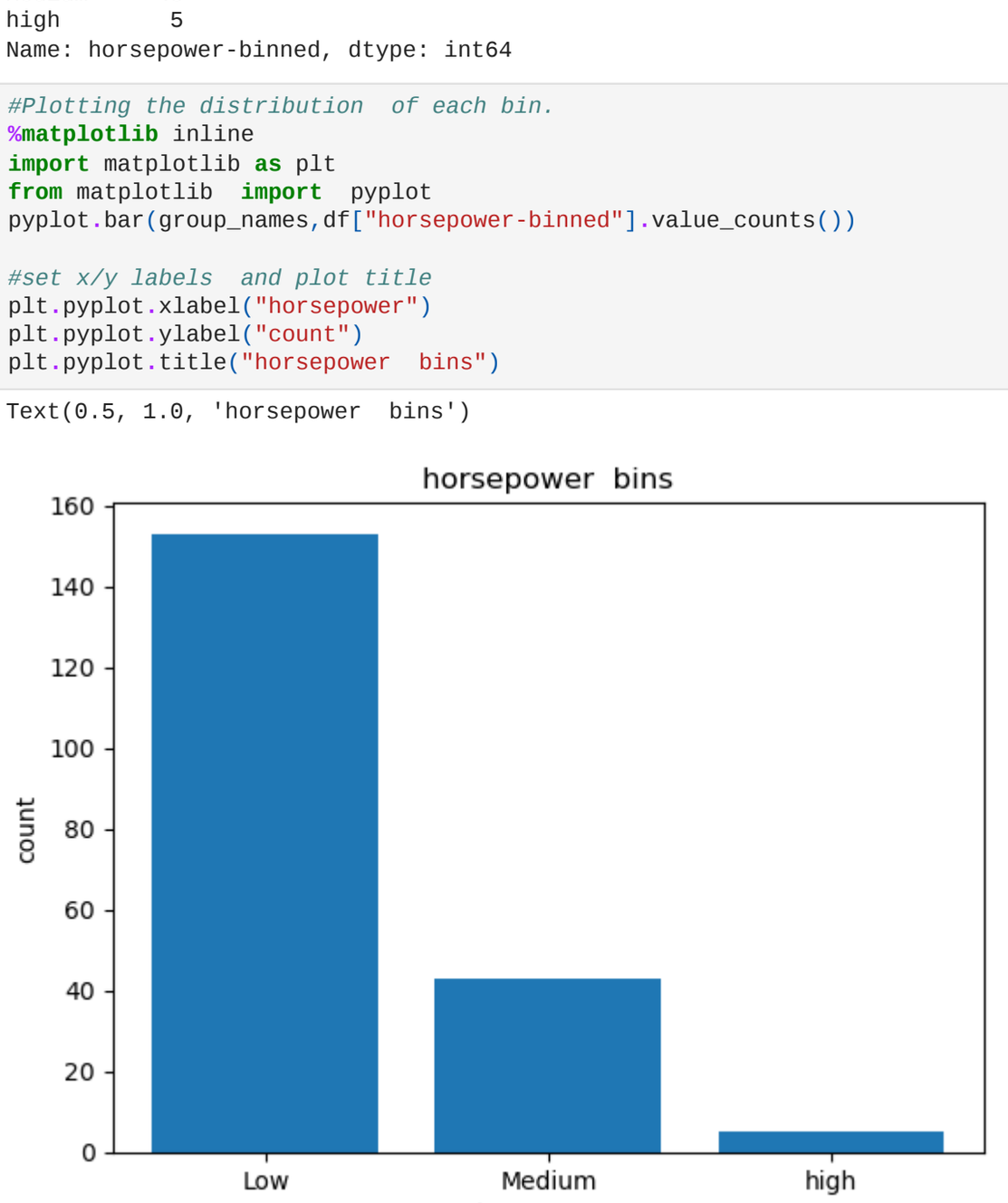
In [49]:
```python
# The number  of vehicles in each bin
df["horsepower-binned"].value_counts()
```

Out[49]:
```
Low       153
Medium     43
high        5
Name: horsepower-binned, dtype: int64
```

In [50]:
```python
#Plotting the distribution  of each bin.
%matplotlib inline
import matplotlib as plt
from matplotlib  import  pyplot
pyplot.bar(group_names,df["horsepower-binned"].value_counts())

#set x/y labels  and plot title
plt.pyplot.xlabel("horsepower")
plt.pyplot.ylabel("count")
plt.pyplot.title("horsepower  bins")
```

Out[50]:
```
Text(0.5, 1.0, 'horsepower  bins')
```



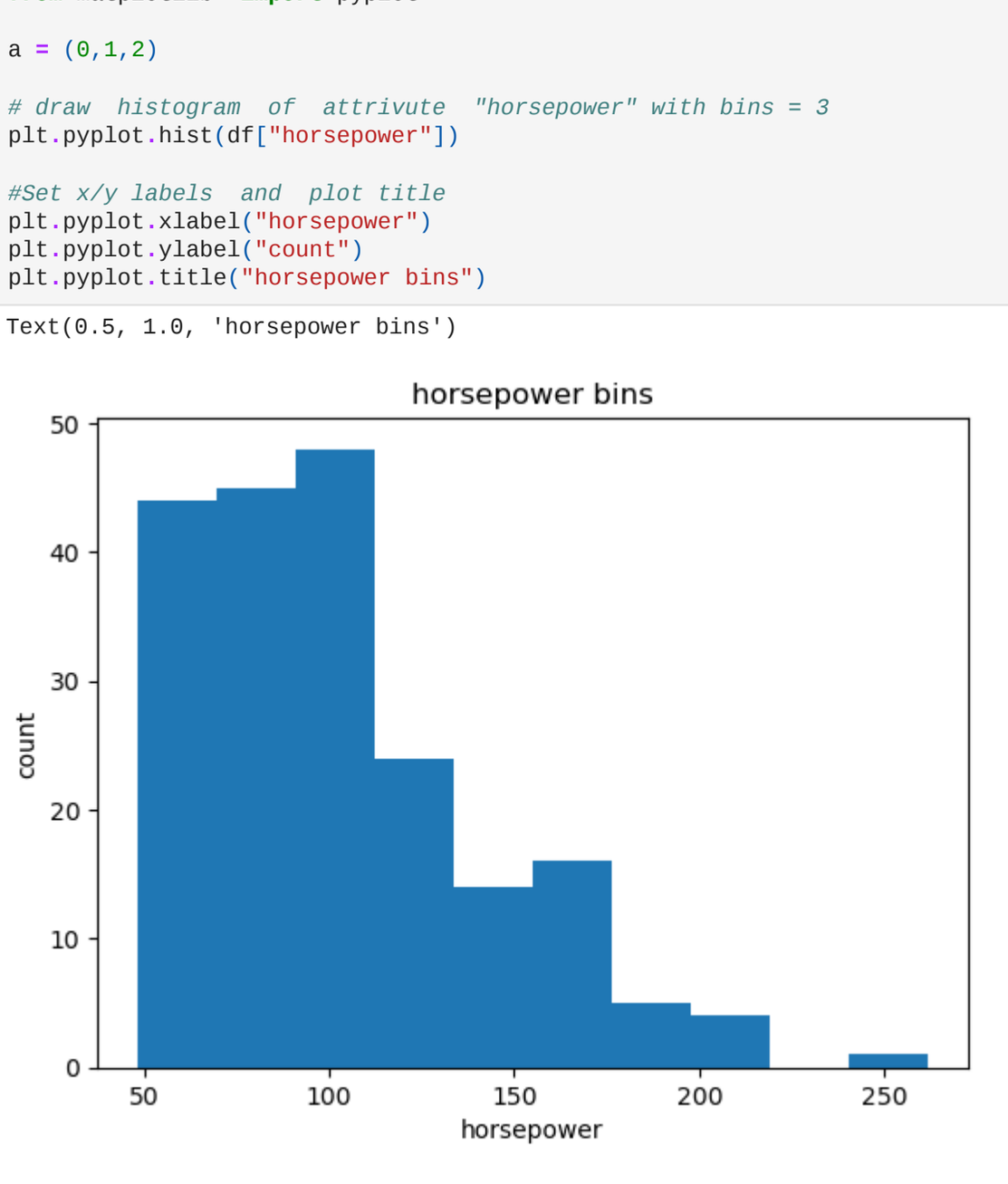
In [51]:
```python
# Histogram is used to visualize  the distribution of  bins we  created above.
%matplotlib inline
import matplotlib  as plt
from matplotlib  import pyplot

a = (0,1,2)

# draw  histogram  of  attrivute  "horsepower" with bins = 3
plt.pyplot.hist(df["horsepower"])

#Set x/y labels  and  plot title
plt.pyplot.xlabel("horsepower")
plt.pyplot.ylabel("count")
plt.pyplot.title("horsepower bins")
```

Out[51]:
```
Text(0.5, 1.0, 'horsepower bins')
```



In [52]:
```python
df.to_csv('clean_df.csv')
```

In [ ]: