In [ ]:

```
Import libraries
```

In [3]:

```
import numpy as np
import pandas as pd
from pandas import DataFrame as df
import matplotlib.pyplot as plt
import seaborn as sns
```

Import dataset store it as dataframe in python

In [4]:

```
filename = "https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/DA0101EN/auto.csv"
```

Add Headers

In [5]:

```
col_headers = ["symboling","normalized-losses","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","compression-ratio","horsepower",
 "peak-rpm","city-mpg","highway-mpg","price"]
```

Read CSV

In [6]:

```
cars = pd.read_csv(filename, names = col_headers)
cars.head(10)
```

Out[6]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	v
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	
3	2	164	audi	gas	std	four	sedan	fwd	front	
4	2	164	audi	gas	std	four	sedan	4wd	front	
5	2	?	audi	gas	std	two	sedan	fwd	front	
6	1	158	audi	gas	std	four	sedan	fwd	front	
7	1	?	audi	gas	std	four	wagon	fwd	front	
8	1	158	audi	gas	turbo	four	sedan	fwd	front	
9	0	?	audi	gas	turbo	two	hatchback	4wd	front	

10 rows × 26 columns

Check for NULL / NaN / NA / '?' values

In [7]:

```
print(cars.isnull())
```

```
     symboling  normalized-losses   make  fuel-type  aspiration  n
um-of-doors  \
0        False              False  False      False       False
False
1        False              False  False      False       False
False
2        False              False  False      False       False
False
3        False              False  False      False       False
False
4        False              False  False      False       False
False
..         ...                ...    ...        ...         ...
...
200      False              False  False      False       False
False
```

```
201      False              False  False      False       False
False
202      False              False  False      False       False
False
203      False              False  False      False       False
False
204      False              False  False      False       False
False

     body-style  drive-wheels  engine-location  wheel-base  ...  e
ngine-size  \
0         False         False            False       False  ...
False
1         False         False            False       False  ...
False
2         False         False            False       False  ...
False
3         False         False            False       False  ...
False
4         False         False            False       False  ...
False
..          ...           ...              ...         ...  ...
...
200       False         False            False       False  ...
False
201       False         False            False       False  ...
False
202       False         False            False       False  ...
False
203       False         False            False       False  ...
False
204       False         False            False       False  ...
False

     fuel-system   bore  stroke  compression-ratio  horsepower  pe
ak-rpm  \
0          False  False   False              False       False
False
1          False  False   False              False       False
False
2          False  False   False              False       False
False
3          False  False   False              False       False
False
4          False  False   False              False       False
False
..           ...    ...     ...                ...         ...
...
200        False  False   False              False       False
False
201        False  False   False              False       False
False
202        False  False   False              False       False
```

```
False
203         False  False   False              False       False
False
204         False  False   False              False       False
False

     city-mpg  highway-mpg  price
0       False        False  False
1       False        False  False
2       False        False  False
3       False        False  False
4       False        False  False
..        ...          ...    ...
200     False        False  False
201     False        False  False
202     False        False  False
203     False        False  False
204     False        False  False

[205 rows x 26 columns]
```

Visible True values indicate presence of NULL '?' valuesReplacing '?' with NaN to improve ease of operations

In [8]:

```
cars = cars.replace('?', np.nan)
```

Count of missing values

In [9]:

```
colna_sum = cars.isnull().sum()
print("Columns with NA values")
print("")
print(colna_sum[colna_sum>0])
print("")
print("Total NA values = {naval_sum:n}".format(naval_sum = cars.isnull().sum()
.sum()))
```

```
Columns with NA values

normalized-losses    41
num-of-doors          2
bore                  4
stroke                4
horsepower            2
peak-rpm              2
price                 4
dtype: int64

Total NA values = 59
```

Dropping missing values of column 'price'

In [10]:

```
cars = cars[cars['price'].notna()]
```

Creating Function to replace NaN values according to their datatypes

In [11]:

```
def replacena_mean(df,k):
    for i in k:
        a = df[i].astype('float').mean(axis =0)
        df[i].replace(np.nan,a,inplace = True)
    return df

def replacena_mode(df,k):
    for i in k:
        df[i].fillna(df[i].mode()[0], inplace = True)
    return df
```

Replacing values

In [12]:

```
cq1 = ['normalized-losses', 'bore', 'stroke', 'horsepower',
       'peak-rpm']
cq2 =  ['num-of-doors']
replacena_mean(cars,cq1)
replacena_mode(cars,cq2)
colna_sum = cars.isnull().sum()
colna_sum[colna_sum>0]
```

Out[12]:

```
Series([], dtype: int64)
```

Resetting the index

In [13]:

```
cars.reset_index()
```

Out[13]:

	index	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels
0	0	3	122	alfa-romero	gas	std	two	convertible	rwd
1	1	3	122	alfa-romero	gas	std	two	convertible	rwd
2	2	1	122	alfa-romero	gas	std	two	hatchback	rwd
3	3	2	164	audi	gas	std	four	sedan	fwd
4	4	2	164	audi	gas	std	four	sedan	4wd
...	...	...	...	...	...	...	...	...	...
196	200	-1	95	volvo	gas	std	four	sedan	rwd
197	201	-1	95	volvo	gas	turbo	four	sedan	rwd
198	202	-1	95	volvo	gas	std	four	sedan	rwd
199	203	-1	95	volvo	diesel	turbo	four	sedan	rwd
200	204	-1	95	volvo	gas	turbo	four	sedan	rwd

201 rows × 27 columns

Checking Datatypes

In [14]:

```
cars.dtypes
```

Out[14]:

```
symboling              int64
normalized-losses     object
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                  object
stroke                object
compression-ratio    float64
horsepower            object
peak-rpm              object
city-mpg               int64
highway-mpg            int64
price                 object
dtype: object
```

We can see that variables are already in Float or Int formatTransforming city-mpg and highway-mpg into liters/100km using conversion formula: L/100km = 235/mpg i.e. creating two new column “city-L/100km” and “highway-L/100km”

In [15]:

```
cars['city-L/100km'] = 235/cars['city-mpg']
cars['highway-L/100km'] = 235/cars['highway-mpg']
cars
```

Out[15]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location
0	3	122	alfa-romero	gas	std	two	convertible	rwd	fron
1	3	122	alfa-romero	gas	std	two	convertible	rwd	fron
2	1	122	alfa-romero	gas	std	two	hatchback	rwd	fron
3	2	164	audi	gas	std	four	sedan	fwd	fron
4	2	164	audi	gas	std	four	sedan	4wd	fron
...	...	...	...	...	...	...	...	...	..
200	-1	95	volvo	gas	std	four	sedan	rwd	fron
201	-1	95	volvo	gas	turbo	four	sedan	rwd	fron
202	-1	95	volvo	gas	std	four	sedan	rwd	fron
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	fron
204	-1	95	volvo	gas	turbo	four	sedan	rwd	fron

201 rows × 28 columns

Normalizing columns length, width, height so that their values range from 0 to 1. i.e. Replacing original values with original_value/max_value

In [16]:

```
cars[['length', 'width', 'height']] = cars[['length', 'width', 'height']]/cars
[['length', 'width', 'height']].max()
cars[['length', 'width', 'height']]
```

Out[16]:

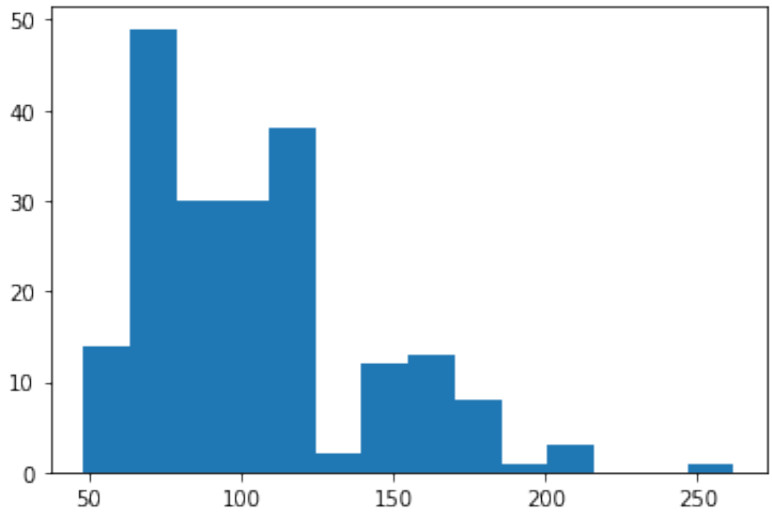
	length	width	height
0	0.811148	0.890278	0.816054
1	0.811148	0.890278	0.816054
2	0.822681	0.909722	0.876254
3	0.848630	0.919444	0.908027
4	0.848630	0.922222	0.908027
...	...	...	...
200	0.907256	0.956944	0.928094
201	0.907256	0.955556	0.928094
202	0.907256	0.956944	0.928094
203	0.907256	0.956944	0.928094
204	0.907256	0.956944	0.928094

201 rows × 3 columns

Plotting the histogram of horsepower to see its distribution

In [25]:

```
cars['horsepower'] = cars['horsepower'].astype('int')
plt.hist(cars.horsepower, bins='auto', rwidth=2, histtype='bar');
```



Creating three equal sized bins “low”, “medium”, “high” and organizing values in column horsepower into

new column “horsepower-binned”

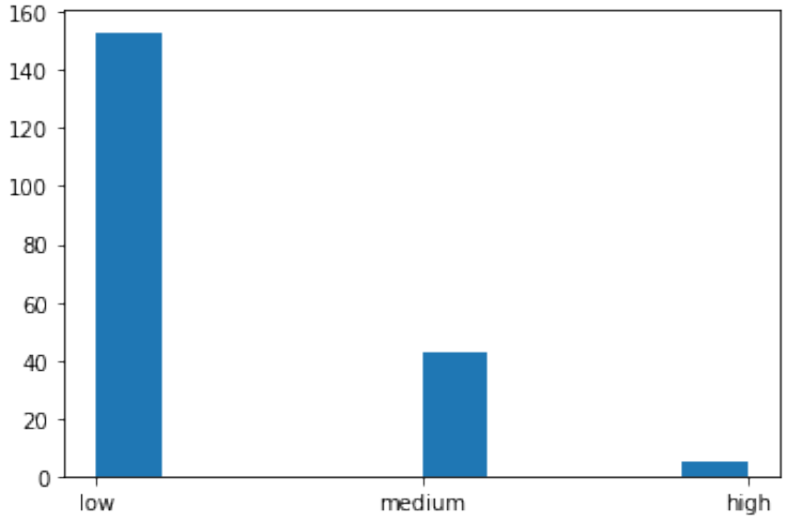
In [26]:

```
cars['horsepower-binned'] = pd.cut(cars['horsepower'], bins = 3, labels = ['lo
w','medium','high'])
```

Plotting distribution of “horsepower-binned”

In [27]:

```
plt.hist(cars['horsepower-binned']);
```



One hot encoding of variables: fuel-type' and 'aspiration'

In [20]:

```
pd.get_dummies(cars[['fuel-type','aspiration']])
pd.get_dummies(cars[['fuel-type','aspiration']]).columns
```

Out[20]:

```
Index(['fuel-type_diesel', 'fuel-type_gas', 'aspiration_std',
       'aspiration_turbo'],
      dtype='object')
```

In [21]:

```
cars[['fuel-type_diesel', 'fuel-type_gas', 'aspiration_std',
       'aspiration_turbo']] = pd.get_dummies(cars[['fuel-type','aspiration']])
```

dropping columns “fuel-type” and “aspiration”

In [22]:

```
cars.drop('fuel-type', axis = 1, inplace=True)
```

In [23]:

```
cars.drop('aspiration', axis = 1, inplace=True)
```

In [24]:

```
cars
```

Out[24]:

	symboling	normalized-losses	make	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length
0	3	122	alfa-romero	two	convertible	rwd	front	88.6	0.811148
1	3	122	alfa-romero	two	convertible	rwd	front	88.6	0.811148
2	1	122	alfa-romero	two	hatchback	rwd	front	94.5	0.822681
3	2	164	audi	four	sedan	fwd	front	99.8	0.848630
4	2	164	audi	four	sedan	4wd	front	99.4	0.848630
...	...	...	...	...	...	...	...	...	..
200	-1	95	volvo	four	sedan	rwd	front	109.1	0.907256
201	-1	95	volvo	four	sedan	rwd	front	109.1	0.907256
202	-1	95	volvo	four	sedan	rwd	front	109.1	0.907256
203	-1	95	volvo	four	sedan	rwd	front	109.1	0.907256
204	-1	95	volvo	four	sedan	rwd	front	109.1	0.907256

201 rows × 31 columns

In [ ]:

```

```

In [ ]:

```

```