```
#Data wrangling

import pandas as pd
df = pd.read_csv('Automobile_data.csv')
df.head()
```

| | symboling | normalized-losses | make | fuel-type | aspiration | num-of-doors | body-style | drive-wheels | engine-location | wheel-base | length | width | height | cu wei |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 3 | ? | alfa-romero | gas | std | two | convertible | rwd | front | 88.6 | 168.8 | 64.1 | 48.8 | 2 |
| **1** | 3 | ? | alfa-romero | gas | std | two | convertible | rwd | front | 88.6 | 168.8 | 64.1 | 48.8 | 2 |
| **2** | 1 | ? | alfa-romero | gas | std | two | hatchback | rwd | front | 94.5 | 171.2 | 65.5 | 52.4 | 2 |
| **3** | 2 | 164 | audi | gas | std | four | sedan | fwd | front | 99.8 | 176.6 | 66.2 | 54.3 | 2 |
| **4** | 2 | 164 | audi | gas | std | four | sedan | 4wd | front | 99.4 | 176.6 | 66.4 | 54.3 | 2 |

```
#Identify and handle missing values
import numpy as np

# replace "?" to NaN
df.replace("?", np.nan, inplace = True)
df.head()
```

| | symboling | normalized-losses | make | fuel-type | aspiration | num-of-doors | body-style | drive-wheels | engine-location | wheel-base | length | width | height | cu wei |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 3 | NaN | alfa-romero | gas | std | two | convertible | rwd | front | 88.6 | 168.8 | 64.1 | 48.8 | 2 |
| **1** | 3 | NaN | alfa-romero | gas | std | two | convertible | rwd | front | 88.6 | 168.8 | 64.1 | 48.8 | 2 |
| **2** | 1 | NaN | alfa- | gas | std | two | hatchback | rwd | front | 94.5 | 171.2 | 65.5 | 52.4 | 2 |

```
# Evaluating Missing values
missing_data = df.isnull()
missing_data.head(5)
```

| | symboling | normalized-losses | make | fuel-type | aspiration | num-of-doors | body-style | drive-wheels | engine-location | wheel-base | length | width | height | curb-weight |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | False | True | 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 |
| **2** | False | True | 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 |
| **4** | False | False | False | False | False | False | False | False | False | False | False | False | False | False |

```
missing_data1 = df.notnull()
missing_data1.head(5)
```

| | symboling | normalized-losses | make | fuel-type | aspiration | num-of-doors | body-style | drive-wheels | engine-location | wheel-base | length | width | height | curb-weight | e |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | True | False | True | True | True | True | True | True | True | True | True | True | True | True | |
| **1** | True | False | True | True | True | True | True | True | True | True | True | True | True | True | |
| **2** | True | False | True | True | True | True | True | True | True | True | True | True | True | True | |

```
df.dtypes
```

```
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
```

```
#Counting missing values in each column
for column in missing_data.columns.values.tolist():
    print(column)
```

```
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
```

```
#Counting 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-losses
False    164
True      41
Name: normalized-losses, 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
```

```
#"normalized-losses","stroke","bore","horsepower","peak-rpm"  replace by mean or median (numeric data)
avg_1 = df["normalized-losses"].astype("float").mean()
avg_1
```

```
    122.0
```

```
df["normalized-losses"].replace(np.nan, avg_1, inplace = True)
```

```
avg_2=df['bore'].astype('float').mean()
avg_2
```

```
    3.3297512437810957
```

```
df['bore'].replace(np.nan, avg_2, inplace= True)
```

```
avg_3 = df['stroke'].astype('float').mean(axis=0)
df['stroke'].replace(np.nan, avg_3, inplace = True)
```

```
avg_4=df['horsepower'].astype('float').mean(axis=0)
```

```
df['horsepower'].replace(np.nan, avg_4, inplace= True)

avg_5=df['peak-rpm'].astype('float').mean(axis=0)
df['peak-rpm'].replace(np.nan, avg_5, inplace= True)
```

```
#replace by mode or maximum occuring frequency
df['num-of-doors'].value_counts()
```

```
four    114
two      89
Name: num-of-doors, dtype: int64
```

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

```
'four'
```

```
#replace the missing 'num-of-doors' values by the most frequent
df["num-of-doors"].replace(np.nan, "four", inplace = True)
```

```
# simply drop whole row with NaN in "price" column
df.dropna(subset=["price"], axis=0, inplace = True)

df.reset_index(drop = True, inplace = True)
```

```
df.head()
```

| | symboling | normalized-losses | make | fuel-type | aspiration | num-of-doors | body-style | drive-wheels | engine-location | wheel-base | length | width | height | cu wei |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 3 | 122 | alfa-romero | gas | std | two | convertible | rwd | front | 88.6 | 168.8 | 64.1 | 48.8 | 2 |
| **1** | 3 | 122 | alfa- | gas | std | two | convertible | rwd | front | 88.6 | 168.8 | 64.1 | 48.8 | 2 |

```
df.dtypes
```

```
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
```

## Data Standardization

```
df['city-l/100km']=235/df["city-mpg"]
```

```
df[["bore", "stroke"]] = df[["bore", "stroke"]].astype("float")
df[["normalized-losses"]] = df[["normalized-losses"]].astype("int")
df[["price"]] = df[["price"]].astype("float")
df[["peak-rpm"]] = df[["peak-rpm"]].astype("float")
```

```
df.dtypes
```

```
#data transformation for highway-mpg into L/100 km
#data normalization :scaling within 1
df['length'] = df['length']/df['length'].max()
df['width'] = df['width']/df['width'].max()
```

```
df['height'] = df['height']/df['height'].max()
df[["length","width","height"]].head()
```

| | 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 |

```
#Binning
df["horsepower"]=df["horsepower"].astype(float)
df["horsepower"]
```

```
0      111.0
1      111.0
2      154.0
3      102.0
4      115.0
       ...  
196    114.0
197    160.0
198    134.0
199    106.0
200    114.0
Name: horsepower, Length: 201, dtype: float64
```

```
binwidth = (max(df["horsepower"])-min(df["horsepower"]))/4
binwidth
```

```
53.5
```

```
bins = np.arange(min(df["horsepower"]), max(df["horsepower"]), binwidth)
bins
```

```
array([ 48. , 101.5, 155. , 208.5])
```

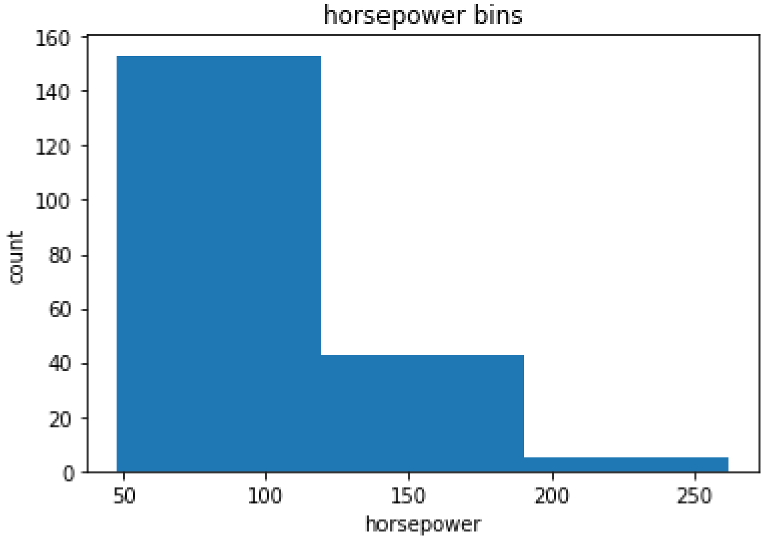
```
group_names = ['Low', 'Medium', 'High']
```

```
df['horsepower-binned'] = pd.cut(df['horsepower'], bins, labels=group_names,include_lowest=True )
df[['horsepower','horsepower-binned']].head(20)
```

| | horsepower | horsepower-binned |
|---|---|---|
| **0** | 111.0 | Medium |
| **1** | 111.0 | Medium |
| **2** | 154.0 | Medium |
| **3** | 102.0 | Medium |
| **4** | 115.0 | Medium |
| **5** | 110.0 | Medium |
| **6** | 110.0 | Medium |
| **7** | 110.0 | Medium |
| **8** | 140.0 | Medium |
| **9** | 101.0 | Low |
| **10** | 101.0 | Low |
| **11** | 121.0 | Medium |
| **12** | 121.0 | Medium |
| **13** | 121.0 | Medium |
| **14** | 182.0 | High |
| **15** | 182.0 | High |
| **16** | 182.0 | High |
| **17** | 48.0 | Low |

```
from matplotlib import pyplot as plt
plt.hist(df["horsepower"], bins = 3)
plt.xlabel("horsepower")
plt.ylabel("count")
plt.title("horsepower bins")
```

```
Text(0.5, 1.0, 'horsepower bins')
```



```
#Indicator variable
df.columns
```

```
Index(['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', 'horsepower-binned'],
      dtype='object')
```

```
dummy_variable_1 = pd.get_dummies(df["fuel-type"])
```

```
dummy_variable_1.rename(columns={'fuel-type-diesel':'gas', 'fuel-type-diesel':'diesel'}, inplace=True)
dummy_variable_1.head()
```

|   | diesel | gas |
|---|---|---|
| **0** | 0 | 1 |
| **1** | 0 | 1 |
| **2** | 0 | 1 |
| **3** | 0 | 1 |

```
df = pd.concat([df, dummy_variable_1], axis=1)
df.drop("fuel-type", axis = 1, inplace=True)
```

```
df.head()
```

|   | symboling | normalized-losses | make | aspiration | num-of-doors | body-style | drive-wheels | engine-location | wheel-base | length | width | height | cur<br>weig |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 3 | 122 | alfa-romero | std | two | convertible | rwd | front | 88.6 | 0.811148 | 0.890278 | 0.816054 | 25 |
| **1** | 3 | 122 | alfa-romero | std | two | convertible | rwd | front | 88.6 | 0.811148 | 0.890278 | 0.816054 | 25 |
| **2** | 1 | 122 | alfa-romero | std | two | hatchback | rwd | front | 94.5 | 0.822681 | 0.909722 | 0.876254 | 28 |
| **3** | 2 | 164 | audi | std | four | sedan | fwd | front | 99.8 | 0.848630 | 0.919444 | 0.908027 | 23 |
| **4** | 2 | 164 | audi | std | four | sedan | 4wd | front | 99.4 | 0.848630 | 0.922222 | 0.908027 | 28 |

```
dummy_variable_2 = pd.get_dummies(df['aspiration'])
dummy_variable_2.rename(columns={'std':'aspiration-std', 'turbo': 'aspiration-turbo'}, inplace=True)
dummy_variable_2.head()
```

| | aspiration-std | aspiration-turbo |
|---|---|---|
| **0** | 1 | 0 |
| **1** | 1 | 0 |
| **2** | 1 | 0 |
| **3** | 1 | 0 |

```
df = pd.concat([df, dummy_variable_2], axis=1)
df.drop('aspiration', axis = 1, inplace=True)
```

```
df.to_csv('clean_df.csv')
```