

Data Wrangling is the process of converting data from the initial format to a format that may be better for analysis.

```
In [1]: import pandas as pd  
import matplotlib.pyplot as plt  
import numpy as np
```

```
In [2]: df = pd.read_csv("/home/hardwarelab00/Downloads/autodata.csv")
```

```
In [3]: df.head(5)
```

```
Out[3]:
```

	Unnamed: 0	symboling	normalized- losses	make	aspiration	num- of- doors	body- style	drive- wheels	engine- location	wheel- base
0	0	3	122	alfa- romero	std	two	convertible	rwd	front	88
1	1	3	122	alfa- romero	std	two	convertible	rwd	front	88
2	2	1	122	alfa- romero	std	two	hatchback	rwd	front	94
3	3	2	164	audi	std	four	sedan	fwd	front	95
4	4	2	164	audi	std	four	sedan	4wd	front	95

5 rows × 30 columns

```
In [4]: df.tail(5)
```

```
Out[4]:
```

	Unnamed: 0	symboling	normalized- losses	make	aspiration	num- of- doors	body- style	drive- wheels	engine- location	wheel- base
196	196	-1	95	volvo	std	four	sedan	rwd	front	109.1
197	197	-1	95	volvo	turbo	four	sedan	rwd	front	109.1
198	198	-1	95	volvo	std	four	sedan	rwd	front	109.1
199	199	-1	95	volvo	turbo	four	sedan	rwd	front	109.1
200	200	-1	95	volvo	turbo	four	sedan	rwd	front	109.1

5 rows × 30 columns

```
In [5]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 201 entries, 0 to 200  
Data columns (total 30 columns):  
 #   Column           Non-Null Count  Dtype    
---  --    
 0   Unnamed: 0        201 non-null    int64   
 1   symboling        201 non-null    int64   
 2   normalized-losses 201 non-null    int64   
 3   make             201 non-null    object   
 4   aspiration       201 non-null    object
```

```

5   num-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  num-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                  197 non-null    float64
20  compression-ratio       201 non-null    float64
21  horsepower                199 non-null    float64
22  peak-rpm                  199 non-null    float64
23  city-mpg                  201 non-null    int64
24  highway-mpg               201 non-null    int64
25  price                      201 non-null    float64
26  city-L/100km              201 non-null    float64
27  horsepower-binned        199 non-null    object
28  diesel                      201 non-null    int64
29  gas                         201 non-null    int64
dtypes: float64(11), int64(9), object(10)
memory usage: 47.2+ KB

```

In [6]: `df.describe()`

	Unnamed: 0	symboling	normalized- losses	wheel- base	length	width	height	curb- weight
count	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000
mean	100.000000	0.840796	122.000000	98.797015	0.837102	0.915126	53.766667	2555.6
std	58.167861	1.254802	31.99625	6.066366	0.059213	0.029187	2.447822	517.2
min	0.000000	-2.000000	65.000000	86.600000	0.678039	0.837500	47.800000	1488.0
25%	50.000000	0.000000	101.000000	94.500000	0.801538	0.890278	52.000000	2169.0
50%	100.000000	1.000000	122.000000	97.000000	0.832292	0.909722	54.100000	2414.0
75%	150.000000	2.000000	137.000000	102.400000	0.881788	0.925000	55.500000	2926.0
max	200.000000	3.000000	256.000000	120.900000	1.000000	1.000000	59.800000	4066.0

In [7]: `df.isnull()`

	Unnamed: 0	symboling	normalized- losses	make	aspiration	num- of- doors	body- style	drive- wheels	engine- location	wheel- base
0	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False

	Unnamed: 0	symboling	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base
...	...	...	...	...	...	...	...	...	...	...
196	False	False	False	False	False	False	False	False	False	False
197	False	False	False	False	False	False	False	False	False	False
198	False	False	False	False	False	False	False	False	False	False
199	False	False	False	False	False	False	False	False	False	False
200	False	False	False	False	False	False	False	False	False	False

201 rows × 30 columns

In [8]: `df.isnull().sum()`

Out[8]:

Unnamed: 0	0
symboling	0
normalized-losses	0
make	0
aspiration	0
num-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
num-of-cylinders	0
engine-size	0
fuel-system	0
bore	0
stroke	4
compression-ratio	0
horsepower	2
peak-rpm	2
city-mpg	0
highway-mpg	0
price	0
city-L/100km	0
horsepower-binned	2
diesel	0
gas	0
dtype:	int64

In [9]: `df.notnull()`

	Unnamed: 0	symboling	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base
0	True	True	True	True	True	True	True	True	True	True
1	True	True	True	True	True	True	True	True	True	True
2	True	True	True	True	True	True	True	True	True	True

	Unnamed: 0	symboling	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base
3	True	True	True	True	True	True	True	True	True	True
4	True	True	True	True	True	True	True	True	True	True
...	...	...	...	...	...	...	...	...	...	...
196	True	True	True	True	True	True	True	True	True	True
197	True	True	True	True	True	True	True	True	True	True
198	True	True	True	True	True	True	True	True	True	True
199	True	True	True	True	True	True	True	True	True	True
200	True	True	True	True	True	True	True	True	True	True

201 rows × 30 columns

In [10]: `df.notnull().sum()`

```
Out[10]: Unnamed: 0      201
symboling      201
normalized-losses  201
make          201
aspiration     201
num-of-doors    201
body-style      201
drive-wheels    201
engine-location  201
wheel-base       201
length          201
width           201
height          201
curb-weight      201
engine-type      201
num-of-cylinders 201
engine-size       201
fuel-system      201
bore            201
stroke          197
compression-ratio 201
horsepower       199
peak-rpm          199
city-mpg          201
highway-mpg        201
price            201
city-L/100km       201
horsepower-binned 199
diesel           201
gas              201
dtype: int64
```

In [54]: `avg_stroke = df["stroke"].astype("float").mean(axis = 0)`  
`print("Average of stroke:", avg_stroke)`  
`df["stroke"].replace(np.nan, avg_stroke, inplace=True)`

Average of stroke: 3.25044027242813

/tmp/ipykernel\_7163/4148721428.py:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inpla

ce method.  
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

```
df["stroke"].replace(np.nan, avg_stroke, inplace=True)
```

```
In [12]: avg_hp = df["horsepower"].astype("float").mean(axis = 0)
print("Average of stroke:", avg_hp)
```

Average of stroke: 103.39698492462311

```
In [55]: df["horsepower"].replace(np.nan, avg_hp, inplace=True)
```

/tmp/ipykernel\_7163/2138043787.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.

The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

```
df["horsepower"].replace(np.nan, avg_hp, inplace=True)
```

```
In [14]: avg_rpm = df["peak-rpm"].astype("float").mean(axis = 0)
print("Average of stroke:", avg_rpm)
```

Average of stroke: 5117.587939698493

```
In [56]: df["peak-rpm"].replace(np.nan, avg_hp, inplace=True)
```

/tmp/ipykernel\_7163/492546639.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.

The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

```
df["peak-rpm"].replace(np.nan, avg_hp, inplace=True)
```

```
In [16]: df['num-of-doors'].value_counts()
```

```
Out[16]: num-of-doors
four      115
two       86
Name: count, dtype: int64
```

```
In [17]: df['num-of-doors'].value_counts().idxmax()
```

```
Out[17]: 'four'
```

```
In [57]: df["num-of-doors"].replace(np.nan, "four", inplace=True)
df.dropna(subset=["horsepower-binned"], axis=0, inplace=True)
df.reset_index(drop=True, inplace=True)
```

/tmp/ipykernel\_7163/1948453182.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.

The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

```
df["num-of-doors"].replace(np.nan, "four", inplace=True)
```

```
In [19]: df.isnull().sum()
```

```
Out[19]: Unnamed: 0      0
symboling          0
normalized-losses  0
make              0
aspiration        0
num-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
num-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
city-L/100km        0
horsepower-binned   0
diesel              0
gas                 0
dtype: int64
```

```
In [20]: df['city-L/100km'] = 235/df["city-mpg"]
df.head()
```

Out[20]:

	Unnamed: 0	symboling	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base
0	0	3	122	alfa-romero	std	two	convertible	rwd	front	88
1	1	3	122	alfa-romero	std	two	convertible	rwd	front	88
2	2	1	122	alfa-romero	std	two	hatchback	rwd	front	94
3	3	2	164	audi	std	four	sedan	fwd	front	95
4	4	2	164	audi	std	four	sedan	4wd	front	95

5 rows × 30 columns

In [21]:

```
df['highway-L/100km'] = 235/df["highway-mpg"]
df.head()
```

Out[21]:

	Unnamed: 0	symboling	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base
0	0	3	122	alfa-romero	std	two	convertible	rwd	front	88
1	1	3	122	alfa-romero	std	two	convertible	rwd	front	88
2	2	1	122	alfa-romero	std	two	hatchback	rwd	front	94
3	3	2	164	audi	std	four	sedan	fwd	front	95
4	4	2	164	audi	std	four	sedan	4wd	front	95

5 rows × 31 columns

In [22]:

```
df['length'] = df['length']/df['length'].max()
df['width'] = df['width']/df['width'].max()
```

In [23]:

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

Out[23]:

	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

In [24]:

```
df.columns
```

Out[24]:

```
Index(['Unnamed: 0', 'symboling', 'normalized-losses', 'make', 'aspiration',
       'num-of-doors', 'body-style', 'drive-wheels', 'engine-location',
```

```
'wheel-base', 'length', 'width', 'height', 'curb-weight', 'engine-type',
'e',
'num-of-cylinders', 'engine-size', 'fuel-system', 'bore', 'stroke',
'compression-ratio', 'horsepower', 'peak-rpm', 'city-mpg',
'highway-mpg', 'price', 'city-L/100km', 'horsepower-binned', 'diesel',
'gas', 'highway-L/100km'],
dtype='object')
```

```
In [25]: df['aspiration'].value_counts()
```

```
Out[25]: aspiration
std      163
turbo     36
Name: count, dtype: int64
```

```
In [26]: dummy_variable_1 = pd.get_dummies(df["aspiration"])
dummy_variable_1.head()
```

```
Out[26]:   std  turbo
0  True  False
1  True  False
2  True  False
3  True  False
4  True  False
```

```
In [27]: df = pd.concat([df, dummy_variable_1], axis=1)
df.drop("aspiration", axis = 1, inplace=True)
```

```
In [28]: df.head()
```

```
Out[28]:   Unnamed: 0  symboling  normalized-losses  make  num-of-doors  body-style  drive-wheels  engine-location  wheel-base  length
0            0          3           122  alfa-romero       two  convertible        rwd      front      88.6  0.811148
1            1          3           122  alfa-romero       two  convertible        rwd      front      88.6  0.811148
2            2          1           122  alfa-romero       two  hatchback        rwd      front      94.5  0.82268
3            3          2           164    audi         four    sedan        fwd      front      99.8  0.848631
4            4          2           164    audi         four    sedan       4wd      front      99.4  0.848631
```

5 rows × 32 columns

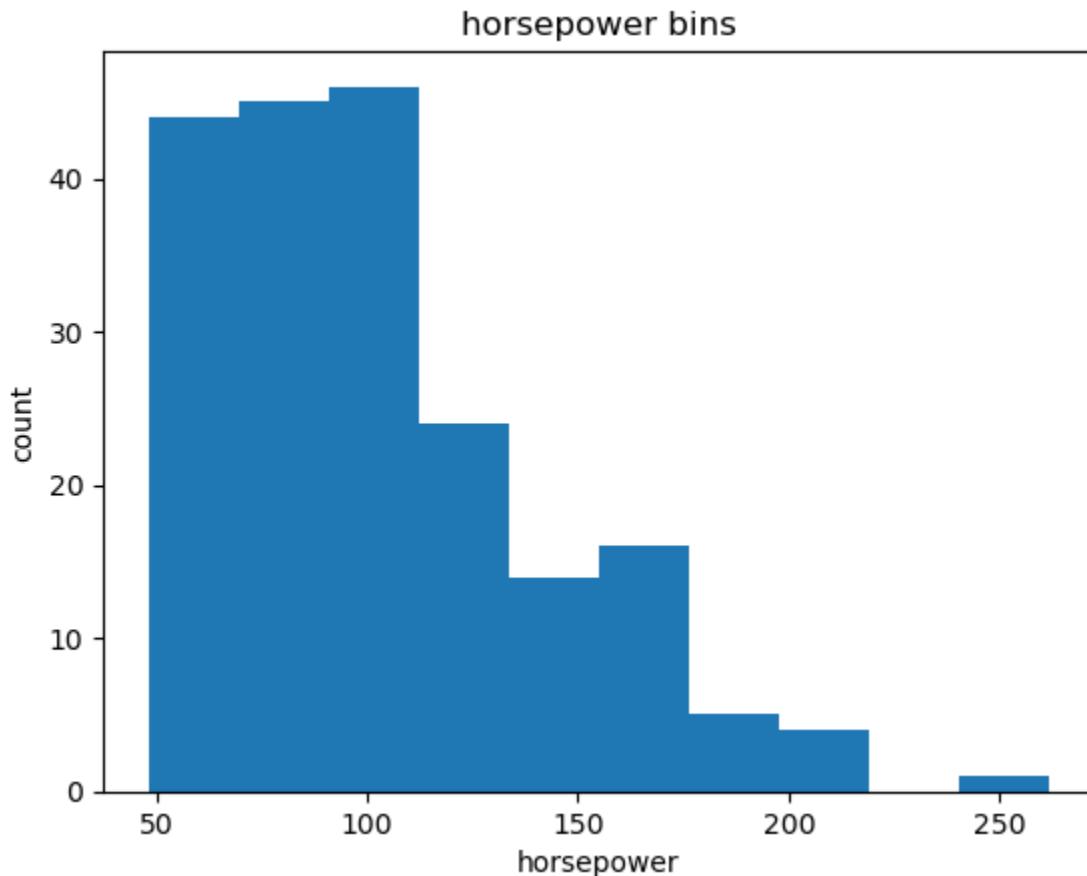
```
In [29]: df["horsepower"] = df["horsepower"].astype(float, copy=True)
```

```
In [30]: %matplotlib inline
import matplotlib as plt
from matplotlib import pyplot
plt.pyplot.hist(df["horsepower"])

plt.pyplot.xlabel("horsepower")
```

```
plt.pyplot.ylabel("count")
plt.pyplot.title("horsepower bins")
```

Out[30]: Text(0.5, 1.0, 'horsepower bins')



In [31]: `bins = np.linspace(min(df["horsepower"]), max(df["horsepower"]), 4)`  
Out[31]: `array([ 48. , 119.33333333, 190.66666667, 262. ])`

In [32]: `group_names = ['Low', 'Medium', 'High']`

In [33]: `df['horsepower-binned'] = pd.cut(df['horsepower'], bins, labels=group_names, include_lowest=True)`  
Out[33]: `df[['horsepower', 'horsepower-binned']].head(20)`

	horsepower	horsepower-binned
0	111.0	Low
1	111.0	Low
2	154.0	Medium
3	102.0	Low
4	115.0	Low
5	110.0	Low
6	110.0	Low
7	110.0	Low
8	140.0	Medium
9	101.0	Low

	horsepower	horsepower-binned
10	101.0	Low
11	121.0	Medium
12	121.0	Medium
13	121.0	Medium
14	182.0	Medium
15	182.0	Medium
16	182.0	Medium
17	48.0	Low
18	70.0	Low
19	70.0	Low

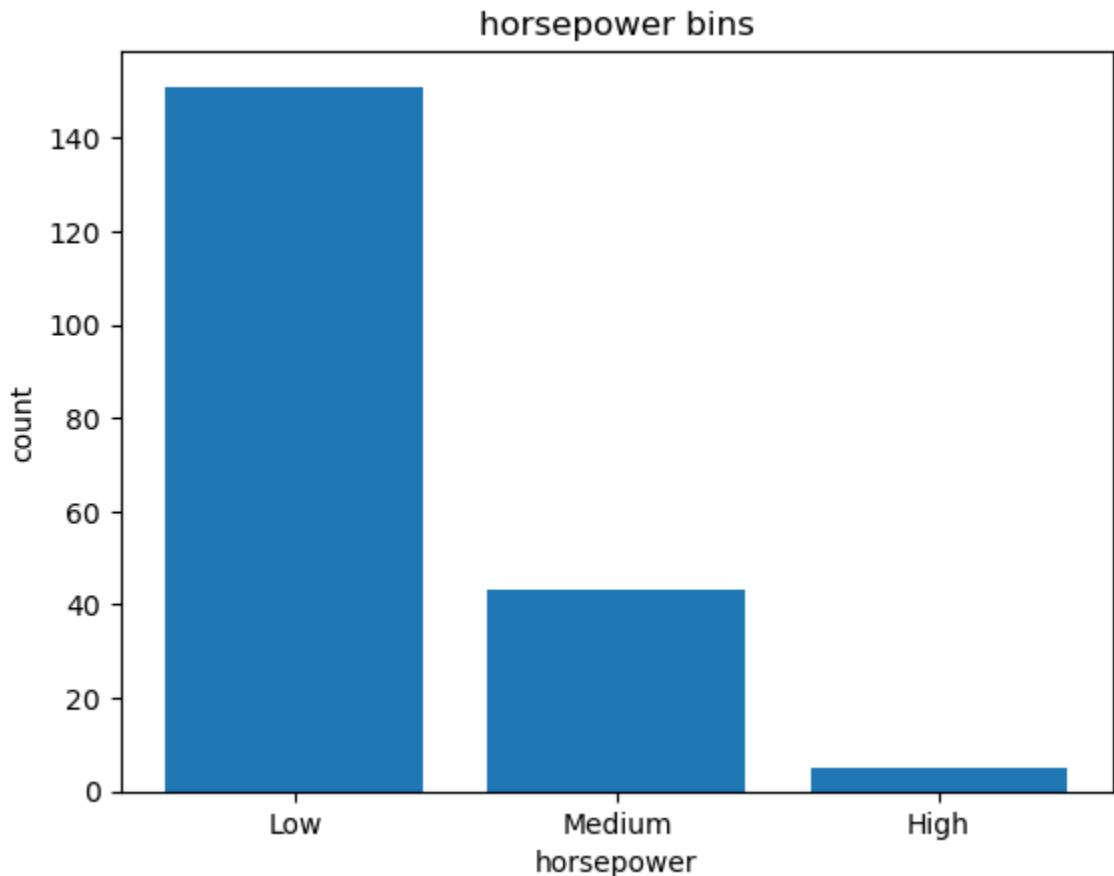
```
In [34]: df["horsepower-binned"].value_counts()
```

```
Out[34]: horsepower-binned
Low      151
Medium    43
High      5
Name: count, dtype: int64
```

```
In [35]: %matplotlib inline
import matplotlib as plt
from matplotlib import pyplot
pyplot.bar(group_names, df["horsepower-binned"].value_counts())

plt.pyplot.xlabel("horsepower")
plt.pyplot.ylabel("count")
plt.pyplot.title("horsepower bins")
```

```
Out[35]: Text(0.5, 1.0, 'horsepower bins')
```

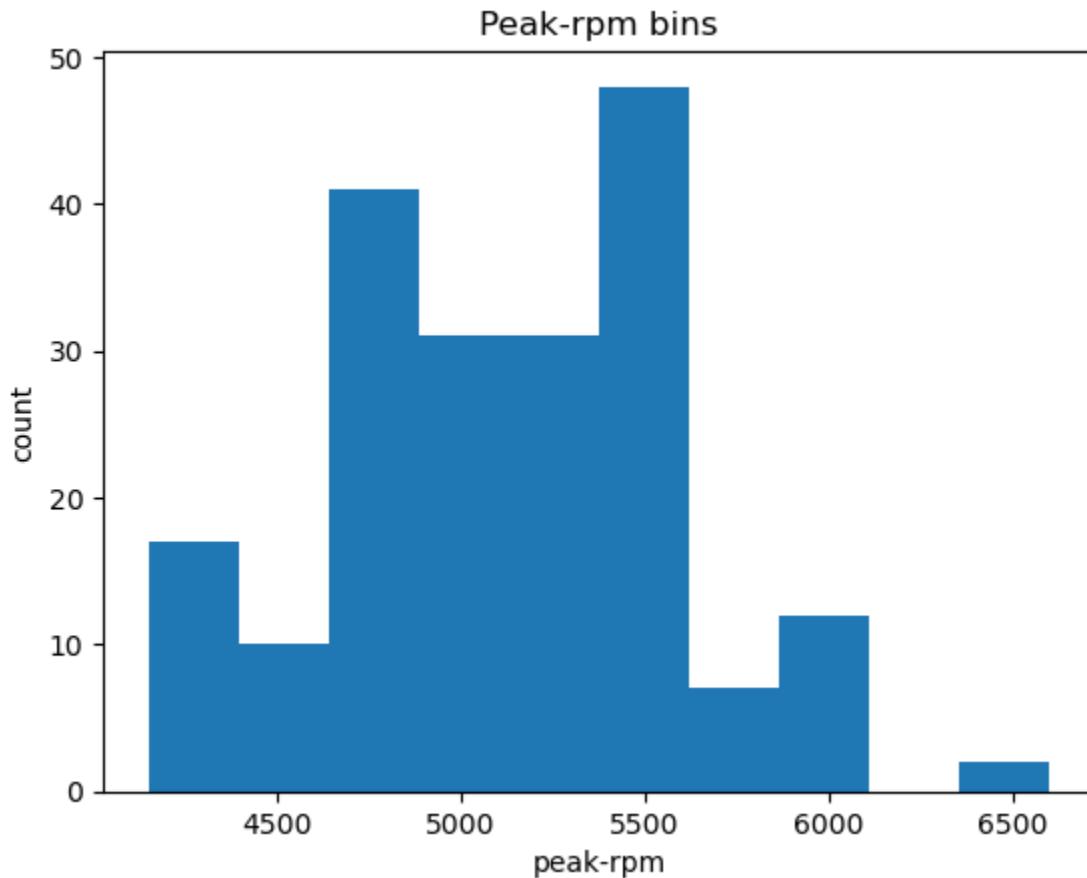


```
In [36]: df["peak-rpm"] = df["peak-rpm"].astype(float, copy=True)
```

```
In [37]: %matplotlib inline
import matplotlib as plt
from matplotlib import pyplot
plt.pyplot.hist(df["peak-rpm"])

plt.pyplot.xlabel("peak-rpm")
plt.pyplot.ylabel("count")
plt.pyplot.title("Peak-rpm bins")
```

```
Out[37]: Text(0.5, 1.0, 'Peak-rpm bins')
```



```
In [38]: bins = np.linspace(min(df["peak-rpm"]), max(df["peak-rpm"]), 4)
bins
```

```
Out[38]: array([4150.0, 4966.66666667, 5783.33333333, 6600.0])
```

```
In [39]: group_names1 = ['Low', 'Medium', 'High']
```

```
In [40]: df['peakrpm-binned'] = pd.cut(df['peak-rpm'], bins, labels=group_names, include_lowest=True)
df[['peak-rpm', 'peakrpm-binned']].head(20)
```

```
Out[40]:   peak-rpm  peakrpm-binned
```

0	5000.0	Medium
1	5000.0	Medium
2	5000.0	Medium
3	5500.0	Medium
4	5500.0	Medium
5	5500.0	Medium
6	5500.0	Medium
7	5500.0	Medium
8	5500.0	Medium
9	5800.0	High
10	5800.0	High
11	4250.0	Low

	peak-rpm	peakrpm-binned
12	4250.0	Low
13	4250.0	Low
14	5400.0	Medium
15	5400.0	Medium
16	5400.0	Medium
17	5100.0	Medium
18	5400.0	Medium
19	5400.0	Medium

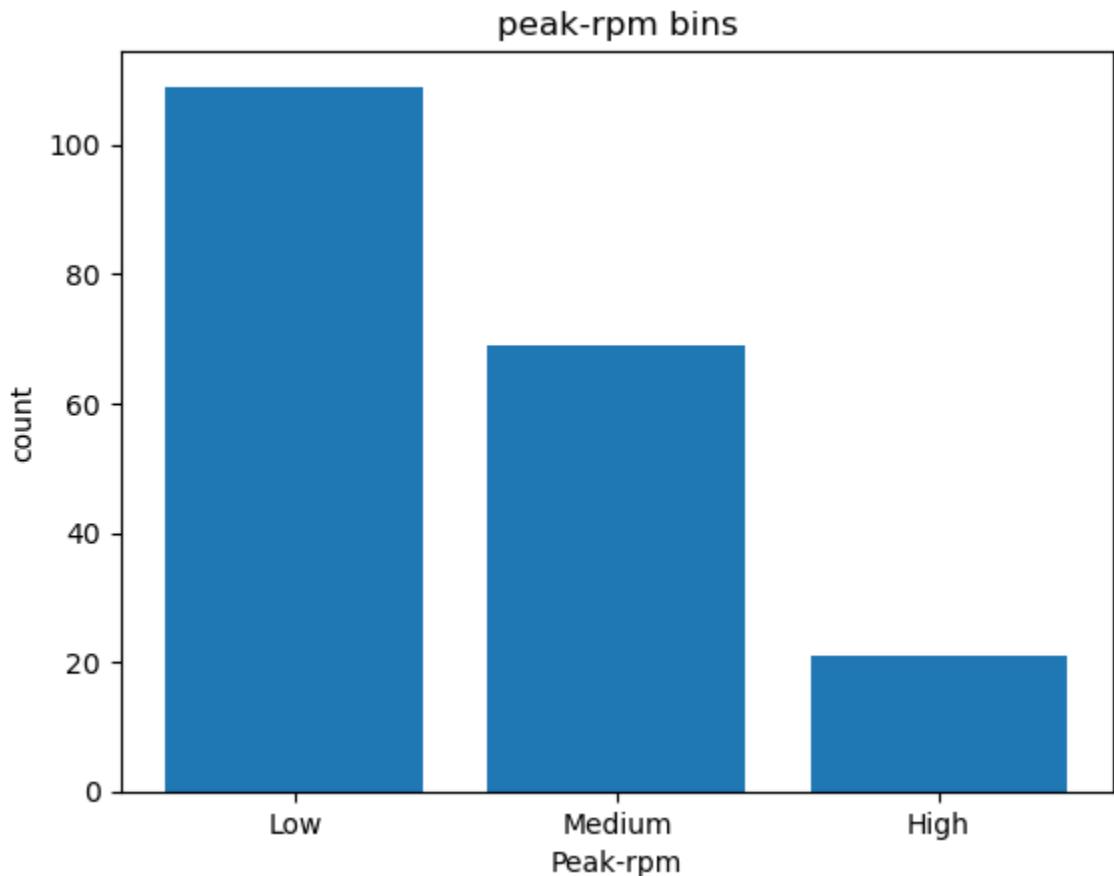
```
In [41]: df["peakrpm-binned"].value_counts()
```

```
Out[41]: peakrpm-binned
Medium    109
Low       69
High      21
Name: count, dtype: int64
```

```
In [42]: %matplotlib inline
import matplotlib as plt
from matplotlib import pyplot
pyplot.bar(group_names, df["peakrpm-binned"].value_counts())

plt.pyplot.xlabel("Peak-rpm")
plt.pyplot.ylabel("count")
plt.pyplot.title("peak-rpm bins")
```

```
Out[42]: Text(0.5, 1.0, 'peak-rpm bins')
```



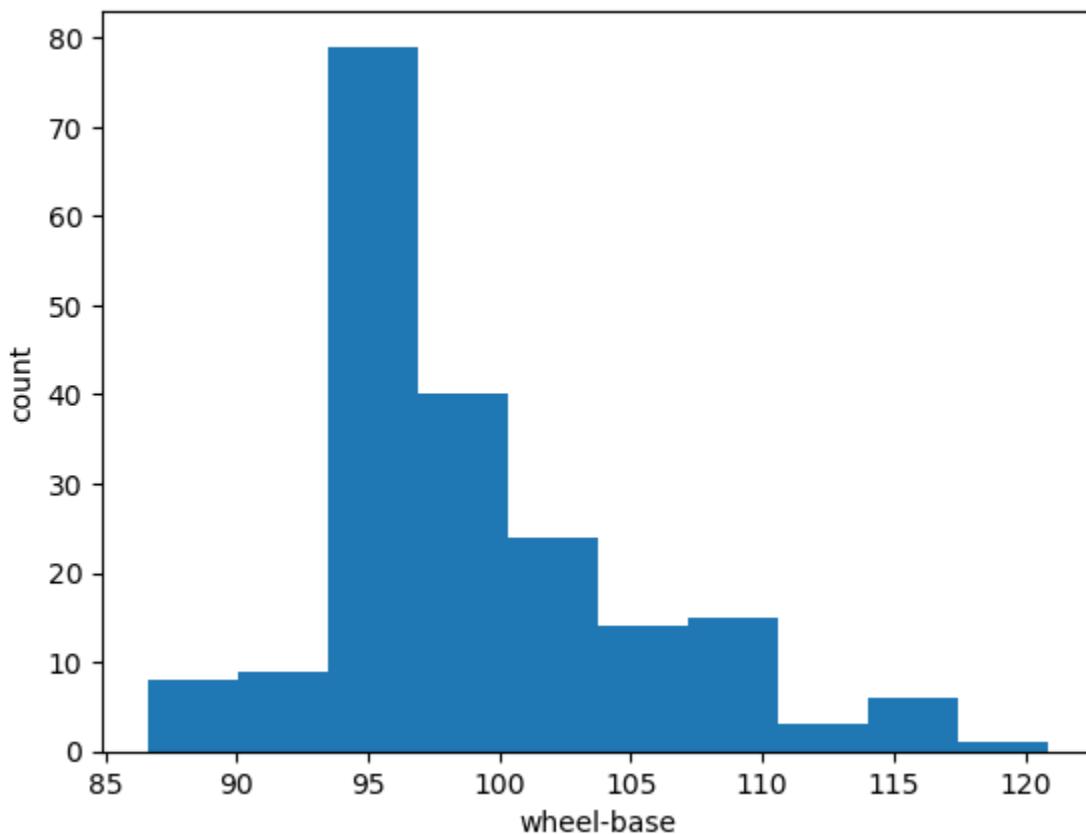
```
In [43]: df["wheel-base"] = df["wheel-base"].astype(float, copy=True)
```

```
In [44]: %matplotlib inline
import matplotlib as plt
from matplotlib import pyplot
plt.pyplot.hist(df["wheel-base"])

plt.pyplot.xlabel("wheel-base")
plt.pyplot.ylabel("count")
plt.pyplot.title("Wheel-base bins")
```

```
Out[44]: Text(0.5, 1.0, 'Wheel-base bins')
```

Wheel-base bins



```
In [45]: bins = np.linspace(min(df["wheel-base"]), max(df["wheel-base"]), 4)
bins
```

```
Out[45]: array([ 86.6           ,  98.03333333, 109.46666667, 120.9           ])
```

```
In [46]: group_names = ['Low', 'Medium', 'High']
```

```
In [47]: df['wheelbase-binned'] = pd.cut(df['wheel-base'], bins, labels=group_names, include_lowest=True)
df[['wheel-base', 'wheelbase-binned']].head(20)
```

```
Out[47]:   wheel-base  wheelbase-binned
```

0	88.6	Low
1	88.6	Low
2	94.5	Low
3	99.8	Medium
4	99.4	Medium
5	99.8	Medium
6	105.8	Medium
7	105.8	Medium
8	105.8	Medium
9	101.2	Medium
10	101.2	Medium
11	101.2	Medium

	wheel-base	wheelbase-binned
12	101.2	Medium
13	103.5	Medium
14	103.5	Medium
15	103.5	Medium
16	110.0	High
17	88.4	Low
18	94.5	Low
19	94.5	Low

```
In [48]: df["wheelbase-binned"].value_counts()
```

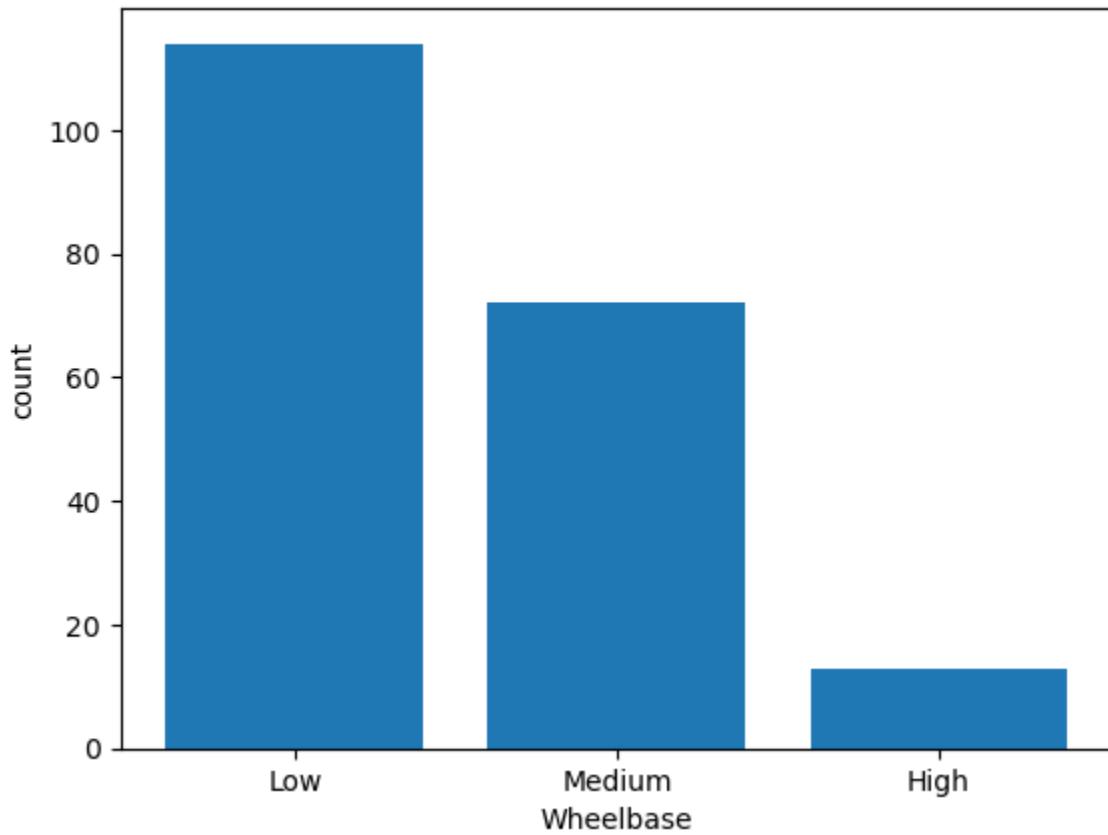
```
Out[48]: wheelbase-binned
```

```
Low      114  
Medium    72  
High     13  
Name: count, dtype: int64
```

```
In [49]: %matplotlib inline  
import matplotlib as plt  
from matplotlib import pyplot  
pyplot.bar(group_names, df["wheelbase-binned"].value_counts())  
  
plt.pyplot.xlabel("Wheelbase")  
plt.pyplot.ylabel("count")  
plt.pyplot.title("Wheelbase bins")
```

```
Out[49]: Text(0.5, 1.0, 'Wheelbase bins')
```

Wheelbase bins



In [ ]: