```
from google.colab import files
files.upload()
```

```
Choose Files No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving clean_df.csv to clean_df.csv
{'clean_df.csv': b',symboling,normalized-losses,make,num-of-doors,body-style,drive-wheel
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv('clean_df.csv')
df.head()
```

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

```
df.describe()
```

	Unnamed: 0	symboling	normalized-losses	wheel-base	length	width	
count	201.000000	201.000000	201.00000	201.000000	201.000000	201.000000	201.(

```
df.dtypes
```

```
Unnamed: 0             int64
symboling              int64
normalized-losses      int64
make                  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
compression-ratio    float64
horsepower           float64
peak-rpm             float64
city-mpg               int64
highway-mpg            int64
Price                float64
city-1/100km         float64
highway-mpg-1        float64
horsepower-binned     object
diesel                 int64
gas                    int64
aspiration-std         int64
aspiration-turbo       int64
dtype: object
```

```
df.shape
```

```
(201, 32)
```

```
df.ndim
```

```
2
```

```
b=df.describe(include=["object"])
b
```

	make	num-of-doors	body-style	drive-wheels	engine-location	engine-type	num-of-cylinders	fuel-system	horsepower-binned
count	201	199	201	201	201	201	201	201	200
unique	22	2	5	3	2	6	7	8	3
top	toyota	four	sedan	fwd	front	ohc	four	mpfi	Low

```
b.shape
```

```
(4, 9)
```

```
df.describe(include=[np.number])
```

	Unnamed: 0	symboling	normalized-losses	wheel-base	length	width	height
count	201.000000	201.000000	201.00000	201.000000	201.000000	201.000000	201.000000
mean	100.000000	0.840796	122.00000	98.797015	0.837102	0.915126	0.899108
std	58.167861	1.254802	31.99625	6.066366	0.059213	0.029187	0.040933
min	0.000000	-2.000000	65.00000	86.600000	0.678039	0.837500	0.799331
25%	50.000000	0.000000	101.00000	94.500000	0.801538	0.890278	0.869565
50%	100.000000	1.000000	122.00000	97.000000	0.832292	0.909722	0.904682
75%	150.000000	2.000000	137.00000	102.400000	0.881788	0.925000	0.928094
max	200.000000	3.000000	256.00000	120.900000	1.000000	1.000000	1.000000

```
a=df.describe(include="all")
a
```

	Unnamed: 0	symboling	normalized-losses	make	num-of-doors	body-style	drive-wheels	engine-location
count	201.000000	201.000000	201.00000	201	199	201	201	201
unique	NaN	NaN	NaN	22	2	5	3	2
top	NaN	NaN	NaN	toyota	four	sedan	fwd	front
freq	NaN	NaN	NaN	32	113	94	118	198

```
a.shape
```

```
(11, 32)
```

```
df.axes
```

```
[RangeIndex(start=0, stop=201, step=1),
 Index(['Unnamed: 0', 'symboling', 'normalized-losses', 'make', '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',
        'city-1/100km', 'highway-mpg-1', 'horsepower-binned', 'diesel', 'gas',
        'aspiration-std', 'aspiration-turbo'],
       dtype='object')]
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201 entries, 0 to 200
Data columns (total 32 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   num-of-doors       199 non-null    object 
 5   body-style         201 non-null    object 
 6   drive-wheels       201 non-null    object 
 7   engine-location    201 non-null    object 
 8   wheel-base         201 non-null    float64
 9   length             201 non-null    float64
 10  width              201 non-null    float64
 11  height             201 non-null    float64
 12  curb-weight        201 non-null    int64  
 13  engine-type        201 non-null    object 
 14  num-of-cylinders   201 non-null    object 
 15  engine-size        201 non-null    int64  
 16  fuel-system        201 non-null    object 
```

```
 17  bore               197 non-null    float64
 18  stroke             201 non-null    float64
 19  compression-ratio  201 non-null    float64
 20  horsepower         201 non-null    float64
 21  peak-rpm           201 non-null    float64
 22  city-mpg           201 non-null    int64
 23  highway-mpg        201 non-null    int64
 24  Price              201 non-null    float64
 25  city-1/100km       201 non-null    float64
 26  highway-mpg-1      201 non-null    float64
 27  horsepower-binned  200 non-null    object
 28  diesel             201 non-null    int64
 29  gas                201 non-null    int64
 30  aspiration-std     201 non-null    int64
 31  aspiration-turbo   201 non-null    int64
dtypes: float64(12), int64(11), object(9)
memory usage: 50.4+ KB
```

```
df.columns
```

```
Index(['Unnamed: 0', 'symboling', 'normalized-losses', 'make', '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',
       'city-1/100km', 'highway-mpg-1', 'horsepower-binned', 'diesel', 'gas',
       'aspiration-std', 'aspiration-turbo'],
      dtype='object')
```

```
df.tail()
```

	Unnamed: 0	symboling	normalized-losses	make	num-of-doors	body-style	drive-wheels	engine-location	wheel bas
196	196	-1	95	volvo	four	sedan	rwd	front	109
197	197	-1	95	volvo	four	sedan	rwd	front	109
198	198	-1	95	volvo	four	sedan	rwd	front	109
199	199	-1	95	volvo	four	sedan	rwd	front	109
200	200	-1	95	volvo	four	sedan	rwd	front	109

```
df['drive-wheels'].value_counts()
```

```
fwd    118
rwd     75
```

```
4wd      8
Name: drive-wheels, dtype: int64
```

```
df['make'].value_counts()
```

```
toyota           32
nissan           18
mazda            17
honda            13
mitsubishi       13
subaru           12
volkswagen       12
peugot           11
volvo            11
dodge             9
mercedes-benz     8
bmw               8
plymouth          7
saab              6
audi              6
porsche           4
jaguar            3
chevrolet         3
alfa-romero       3
isuzu             2
renault           2
mercury           1
Name: make, dtype: int64
```

```
drive_wheels_counts = df['drive-wheels'].value_counts().to_frame()
drive_wheels_counts.rename(columns={'drive-wheels': 'value_counts'}, inplace=True)
drive_wheels_counts
```

	value_counts
fwd	118
rwd	75
4wd	8

```
#give name to index
drive_wheels_counts.index.name="wheels"
drive_wheels_counts
```

```
df
```

	Unnamed: 0	symboling	normalized-losses	make	num-of-doors	body-style	value_counts	en loc
0	0	3	122	alfa-romero	two	convertible	rwd	
1	1	3	122	alfa-romero	two	convertible	rwd	
2	2	1	122	alfa-romero	two	hatchback	rwd	
3	3	2	164	audi	four	sedan	fwd	
4	4	2	164	audi	four	sedan	4wd	
...	...	...	...	...	...	...	...	...
196	196	-1	95	volvo	four	sedan	rwd	
197	197	-1	95	volvo	four	sedan	rwd	
198	198	-1	95	volvo	four	sedan	rwd	
199	199	-1	95	volvo	four	sedan	rwd	
200	200	-1	95	volvo	four	sedan	rwd	

201 rows × 32 columns

```
df.rename(columns={'value_counts':'drive-wheels'},inplace=True)
df
```

	Unnamed: 0	symboling	normalized-losses	make	num-of-doors	body-style	drive-wheels	engine-location	wheel bas
0	0	3	122	alfa-romero	two	convertible	rwd	front	88.
1	1	3	122	alfa-romero	two	convertible	rwd	front	88.
2	2	1	122	alfa-romero	two	hatchback	rwd	front	94.

```
df['drive-wheels'].unique()
```

```
array(['rwd', 'fwd', '4wd'], dtype=object)
```

	...	...	...	...	...	...	...	...	.

```
df_group_one=df[['drive-wheels','body-style','Price']]
df_group_one
```

	drive-wheels	body-style	Price
0	rwd	convertible	13495.0
1	rwd	convertible	16500.0
2	rwd	hatchback	16500.0
3	fwd	sedan	13950.0
4	4wd	sedan	17450.0
...	...	...	...
196	rwd	sedan	16845.0
197	rwd	sedan	19045.0
198	rwd	sedan	21485.0
199	rwd	sedan	22470.0
200	rwd	sedan	22625.0

201 rows × 3 columns

Groupby

```
df_group_one=df[['drive-wheels','body-style','Price']]
df_group_one=df_group_one.groupby(['drive-wheels'],as_index= False).mean()
df_group_one
```

	drive-wheels	Price
0	4wd	10241.000000
1	fwd	9244.779661
2	rwd	19757.613333

```
df_gptest=df[['drive-wheels','body-style','Price']]
grouped_test1=df_gptest.groupby(['drive-wheels','body-style'],as_index= False).mean()
grouped_test1
```

	drive-wheels	body-style	Price
0	4wd	hatchback	7603.000000
1	4wd	sedan	12647.333333
2	4wd	wagon	9095.750000
3	fwd	convertible	11595.000000
4	fwd	hardtop	8249.000000
5	fwd	hatchback	8396.387755
6	fwd	sedan	9811.800000
7	fwd	wagon	9997.333333
8	rwd	convertible	23949.600000
9	rwd	hardtop	24202.714286
10	rwd	hatchback	14337.777778
11	rwd	sedan	21711.833333
12	rwd	wagon	16994.222222

```
# Use the "groupby" function to find the average "price" of each car based on "body-style" ?
df_group_body_price = df[['Price', 'body-style']]
df_group_3 =df_group_body_price.groupby(['body-style'],as_index= False).mean()
df_group_3
```

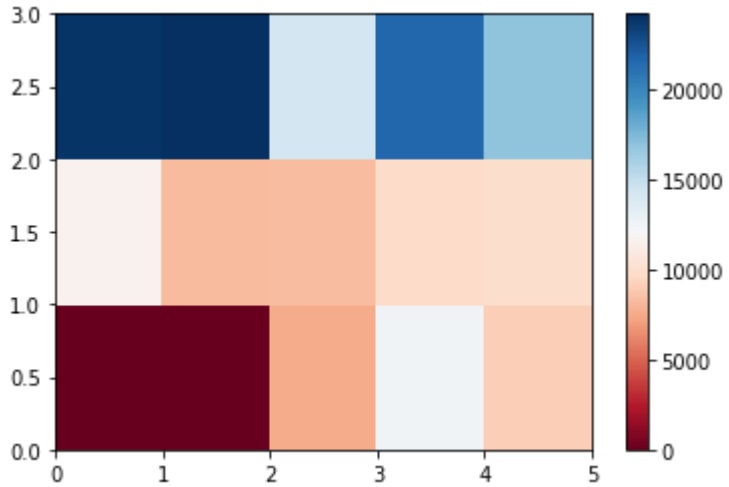
```
grouped_pivot=grouped_test1.pivot(index='drive-wheels',columns='body-style')
grouped_pivot
```

	Price				
body-style	convertible	hardtop	hatchback	sedan	wagon
drive-wheels					
4wd	NaN	NaN	7603.000000	12647.333333	9095.750000
fwd	11595.0	8249.000000	8396.387755	9811.800000	9997.333333
rwd	23949.6	24202.714286	14337.777778	21711.833333	16994.222222

```
grouped_pivot=grouped_pivot.fillna(0) #fill missing values with 0
grouped_pivot
```

	Price				
body-style	convertible	hardtop	hatchback	sedan	wagon
drive-wheels					
4wd	0.0	0.000000	7603.000000	12647.333333	9095.750000
fwd	11595.0	8249.000000	8396.387755	9811.800000	9997.333333
rwd	23949.6	24202.714286	14337.777778	21711.833333	16994.222222

```
plt.pcolor(grouped_pivot,cmap='RdBu')
plt.colorbar()
plt.show()
```




```
# Numeric variables    Non_Graphical : Correlation lies between (-1 to 1)
```

```
#
df.corr()
```

	Unnamed: 0	symboling	normalized-losses	wheel-base	length	width	h
Unnamed: 0	1.000000	-0.162764	-0.241092	0.125517	0.161848	0.043976	0.2
symboling	-0.162764	1.000000	0.466264	-0.535987	-0.365404	-0.242423	-0.5
normalized-losses	-0.241092	0.466264	1.000000	-0.056661	0.019424	0.086802	-0.3
wheel-base	0.125517	-0.535987	-0.056661	1.000000	0.876024	0.814507	0.5
length	0.161848	-0.365404	0.019424	0.876024	1.000000	0.857170	0.4
width	0.043976	-0.242423	0.086802	0.814507	0.857170	1.000000	0.3
height	0.252015	-0.550160	-0.373737	0.590742	0.492063	0.306002	1.0
curb-weight	0.064820	-0.233118	0.099404	0.782097	0.880665	0.866201	0.3
engine-size	-0.047764	-0.110581	0.112360	0.572027	0.685025	0.729436	0.0
bore	0.246289	-0.144324	-0.030035	0.494884	0.610051	0.544924	0.1
stroke	-0.162490	-0.008153	0.055045	0.158018	0.123952	0.188822	-0.0
compression-ratio	0.144301	-0.182196	-0.114713	0.250313	0.159733	0.189867	0.2
horsepower	-0.022474	0.075819	0.217299	0.371147	0.579821	0.615077	-0.0
peak-rpm	-0.195662	0.279740	0.239543	-0.360305	-0.285970	-0.245800	-0.3
city-mpg	0.027956	-0.035527	-0.225016	-0.470606	-0.665192	-0.633531	-0.0
highway-mpg	0.020344	0.036233	-0.181877	-0.543304	-0.698142	-0.680635	-0.1
Price	-0.118214	-0.082391	0.133999	0.584642	0.690628	0.751265	0.1
city-1/100km	-0.099157	0.066171	0.238567	0.476153	0.657373	0.673363	0.0
highway-mpg-1	-0.078346	-0.029807	0.181189	0.577576	0.707108	0.736728	0.0
diesel	0.121454	-0.196735	-0.101546	0.307237	0.211187	0.244356	0.2
gas	-0.121454	0.196735	0.101546	-0.307237	-0.211187	-0.244356	-0.2
aspiration-std	-0.082739	0.054615	0.006911	-0.256889	-0.230085	-0.305732	-0.0
aspiration-turbo	0.082739	-0.054615	-0.006911	0.256889	0.230085	0.305732	0.0

```
df[["wheel-base",'Price']].corr()
```

	wheel-base	Price
wheel-base	1.000000	0.584642
Price	0.584642	1.000000

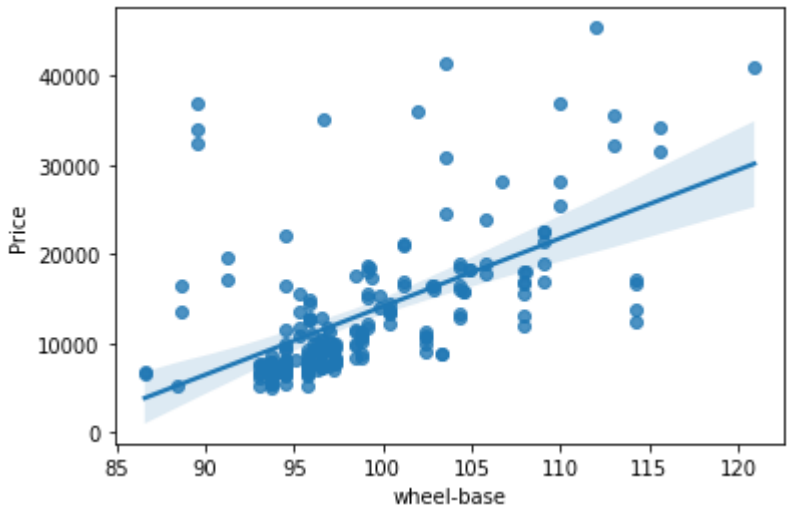
```
df[["engine-size","Price","horsepower"]].corr()
```

	engine-size	Price	horsepower
engine-size	1.000000	0.872335	0.822676
Price	0.872335	1.000000	0.809575
horsepower	0.822676	0.809575	1.000000

```
# for graphical we can use scatterplot
```

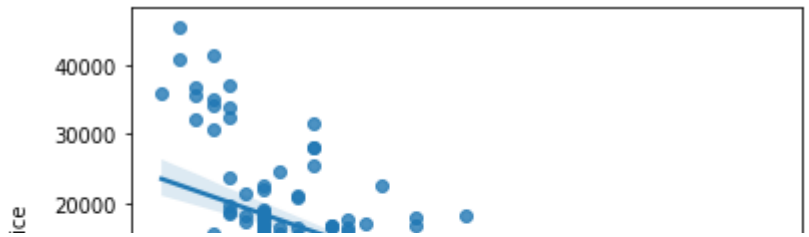
```
sns.regplot(x="wheel-base",y="Price",data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fed0f452b50>
```




```
sns.regplot(x="city-mpg",y="Price",data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fed0f3e4250>
```




```
sns.boxplot(x="drive-wheels",y="Price",data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fed0ee4d090>
```