In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy as sp
```

In [2]:

```
# using the Csv file
df = pd.read_csv('output.csv')

# Checking the first 5 entries of dataset
df.head()
```

Out[2]:

	Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9	Column10	...	Column17	Column18	C
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	

5 rows × 26 columns

In [3]:

```
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"]

df.columns=headers
df.head()
```

Out[3]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	c
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	

5 rows × 26 columns

In [6]:

```
df.dropna(subset=["price"], axis=0, inplace=True)
df=df.dropna(subset=["price"], axis=0)
```

In [11]:

```
df.head()
```

Out[11]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	c
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	

5 rows × 26 columns

In [8]:

```
data = df

# Finding the missing values
data.isna().any()

# Finding if missing values
data.isnull().any()
```

Out[8]:

```
symboling            False
normalized-losses    False
make                 False
fuel-type            False
aspiration           False
num-of-doors         False
body-style           False
drive-wheels         False
engine-location      False
wheel-base           False
length               False
width                False
height               False
curb-weight          False
engine-type          False
num-of-cylinders     False
engine-size          False
fuel-system          False
bore                 False
stroke               False
compression-ratio    False
horsepower           False
peak-rpm             False
city-mpg             False
highway-mpg          False
price                False
dtype: bool
```

In [12]:

```
df["price"].tail(5)
```

Out[12]:

```
200    16845
201    19045
202    21485
203    22470
204    22625
Name: price, dtype: object
```

In [20]:

```
data.price.unique()
```

Out[20]:

```
array(['13495', '16500', '13950', '17450', '15250', '17710', '18920',
       '23875', '?', '16430', '16925', '20970', '21105', '24565', '30760',
       '41315', '36880', '5151', '6295', '6575', '5572', '6377', '7957',
       '6229', '6692', '7609', '8558', '8921', '12964', '6479', '6855',
       '5399', '6529', '7129', '7295', '7895', '9095', '8845', '10295',
       '12945', '10345', '6785', '11048', '32250', '35550', '36000',
       '5195', '6095', '6795', '6695', '7395', '10945', '11845', '13645',
       '15645', '8495', '10595', '10245', '10795', '11245', '18280',
       '18344', '25552', '28248', '28176', '31600', '34184', '35056',
       '40960', '45400', '16503', '5389', '6189', '6669', '7689', '9959',
       '8499', '12629', '14869', '14489', '6989', '8189', '9279', '5499',
       '7099', '6649', '6849', '7349', '7299', '7799', '7499', '7999',
       '8249', '8949', '9549', '13499', '14399', '17199', '19699',
       '18399', '11900', '13200', '12440', '13860', '15580', '16900',
       '16695', '17075', '16630', '17950', '18150', '12764', '22018',
       '32528', '34028', '37028', '9295', '9895', '11850', '12170',
       '15040', '15510', '18620', '5118', '7053', '7603', '7126', '7775',
       '9960', '9233', '11259', '7463', '10198', '8013', '11694', '5348',
       '6338', '6488', '6918', '7898', '8778', '6938', '7198', '7788',
       '7738', '8358', '9258', '8058', '8238', '9298', '9538', '8449',
       '9639', '9989', '11199', '11549', '17669', '8948', '10698', '9988',
       '10898', '11248', '16558', '15998', '15690', '15750', '7975',
       '7995', '8195', '9495', '9995', '11595', '9980', '13295', '13845',
       '12290', '12940', '13415', '15985', '16515', '18420', '18950',
       '16845', '19045', '21485', '22470', '22625'], dtype=object)
```

In [21]:

```
# Here it contains '?', so we Drop it
data = data[data.price != '?']
```

In [22]:

```
# checking it again
data.dtypes
```

Out[22]:

```
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
```

In [23]:

```
data["price"] = data["price"].astype(str).astype(int)
```

```
<ipython-input-23-1d3335603012>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexi
ng.html#returning-a-view-versus-a-copy
  data["price"] = data["price"].astype(str).astype(int)
```

In [24]:

```
# checking it again
data.dtypes
```

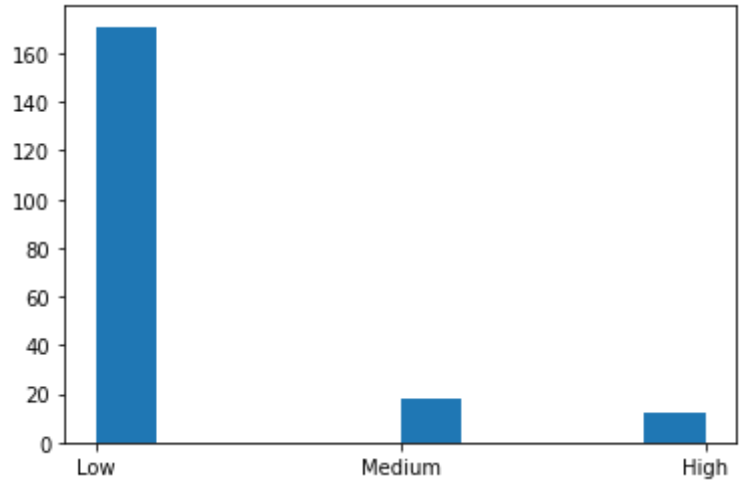
Out[24]:

```
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                  int64
dtype: object
```

In [28]:

```
data['length'] = data['length']/data['length'].max()
data['width'] = data['width']/data['width'].max()
data['height'] = data['height']/data['height'].max()
#convert price to int
data["price"] = data["price"].astype(str).astype(int)
# binning- grouping values
bins = np.linspace(min(data['price']), max(data['price']), 4)
group_names = ['Low', 'Medium', 'High']
data['price-binned'] = pd.cut(data['price'], bins,
                              labels = group_names,
                              include_lowest = True)
print(data['price-binned'])
plt.hist(data['price-binned'])
plt.show()
```

```
0         Low
1         Low
2         Low
3         Low
4         Low
        ...
200       Low
201    Medium
202    Medium
203    Medium
204    Medium
Name: price-binned, Length: 201, dtype: category
Categories (3, object): [Low < Medium < High]
```


160
140
120
100
80
60
40
20
0
Low
Medium
High

In [30]:

```
df.describe()
```

Out[30]:

	symboling	wheel-base	length	width	height	curb-weight	engine-size	compression-ratio	city-mpg	highway-mpg
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	0.834146	98.756585	0.836373	0.911588	0.898409	2555.565854	126.907317	10.142537	25.219512	30.751220
std	1.245307	6.021776	0.059285	0.029671	0.040862	520.680204	41.642693	3.972040	6.542142	6.886443
min	-2.000000	86.600000	0.678039	0.834025	0.799331	1488.000000	61.000000	7.000000	13.000000	16.000000
25%	0.000000	94.500000	0.799135	0.886584	0.869565	2145.000000	97.000000	8.600000	19.000000	25.000000
50%	1.000000	97.000000	0.832292	0.905947	0.904682	2414.000000	120.000000	9.000000	24.000000	30.000000
75%	2.000000	102.400000	0.879865	0.925311	0.928094	2935.000000	141.000000	9.400000	30.000000	34.000000
max	3.000000	120.900000	1.000000	1.000000	1.000000	4066.000000	326.000000	23.000000	49.000000	54.000000

In [37]:

```
drive_wheels_counts=df["drive-wheels"].value_counts()
drive_wheels_counts.rename(colunms=('drive-wheels':'value_counts' inplace = True)
drive_wheels_counts.index.name= 'drive-wheels'
```

```
  File "<ipython-input-37-0d302f96b64b>", line 2
    drive_wheels_counts.rename(colunms=('drive-wheels':'value_counts' inplace = True)
                                                      ^
SyntaxError: invalid syntax
```

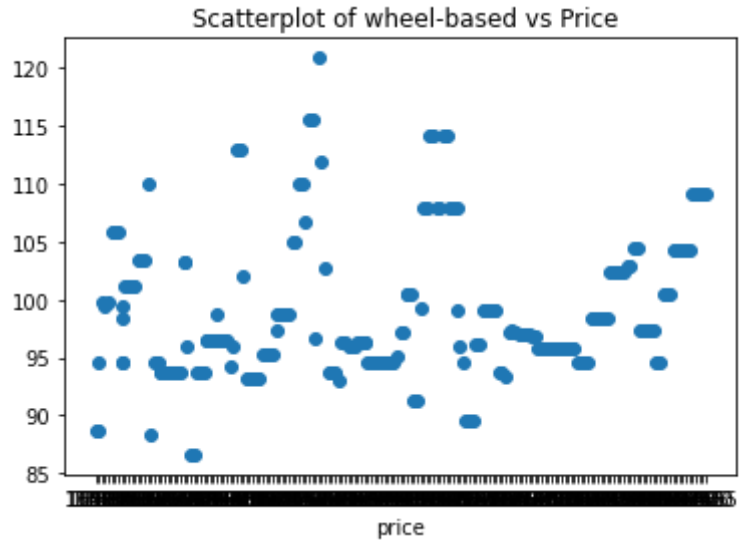
In [41]:

```
y=df["wheel-base"]
x=df["price"]
plt.scatter(x,y)

plt.title (" Scatterplot of wheel-based vs Price")
plt.xlabel("wheel-base")
plt.xlabel("price")
```

Out[41]:

```
Text(0.5, 0, 'price')
```



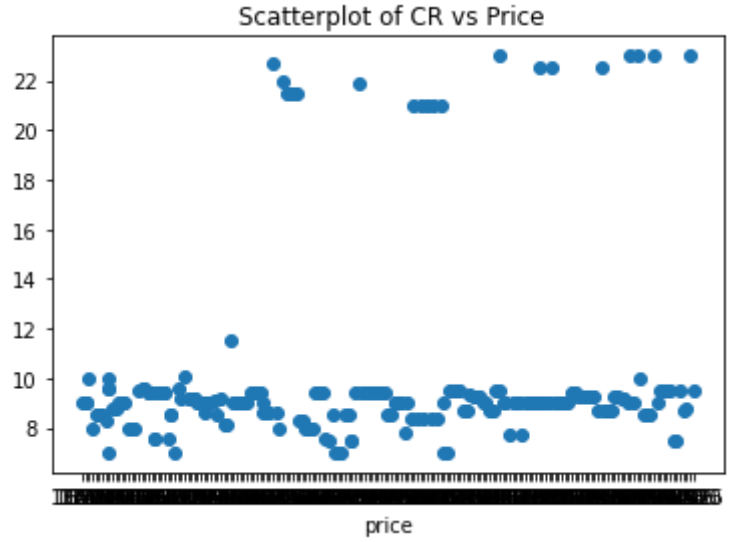
In [42]:

```
y=df["compression-ratio"]
x=df["price"]
plt.scatter(x,y)

plt.title (" Scatterplot of CR vs Price")
plt.xlabel("compression-ratio")
plt.xlabel("price")
```

Out[42]:

```
Text(0.5, 0, 'price')
```

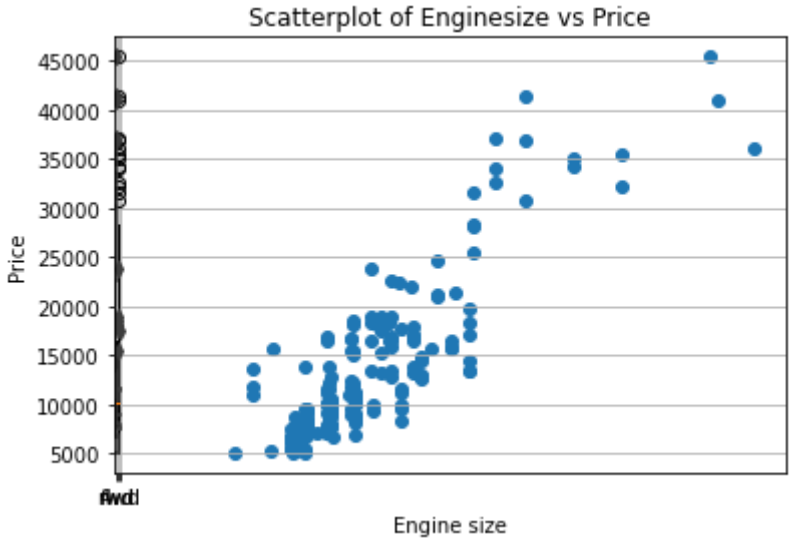


In [56]:

```
# examples of box plot
plt.boxplot(data['price'])

# by using seaborn
sns.boxplot(x ='drive-wheels', y ='price', data = data)

# Predicting price based on engine size
# Known on x and predictable on y
plt.scatter(data['engine-size'], data['price'])
plt.title('Scatterplot of Enginesize vs Price')
plt.xlabel('Engine size')
plt.ylabel('Price')
plt.grid()
plt.show()
```

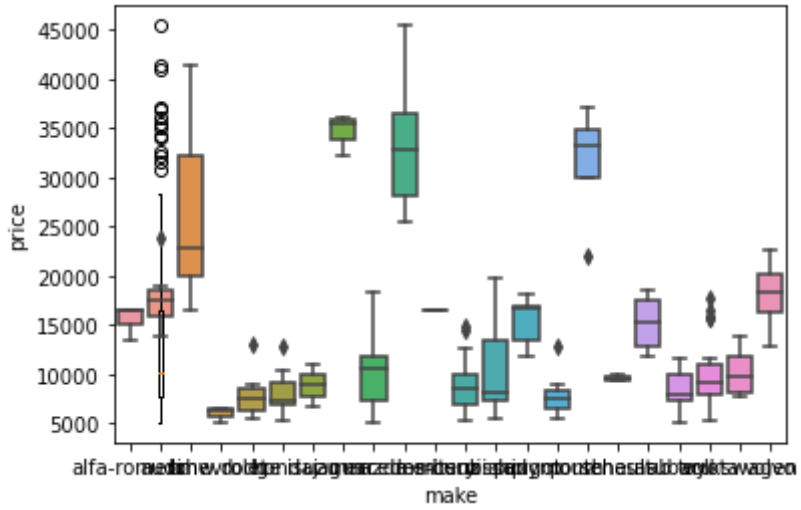


In [58]:

```
# examples of box plot
plt.boxplot(data['price'])

# by using seaborn
sns.boxplot(x ='make', y ='price', data = data)

plt.show()
```

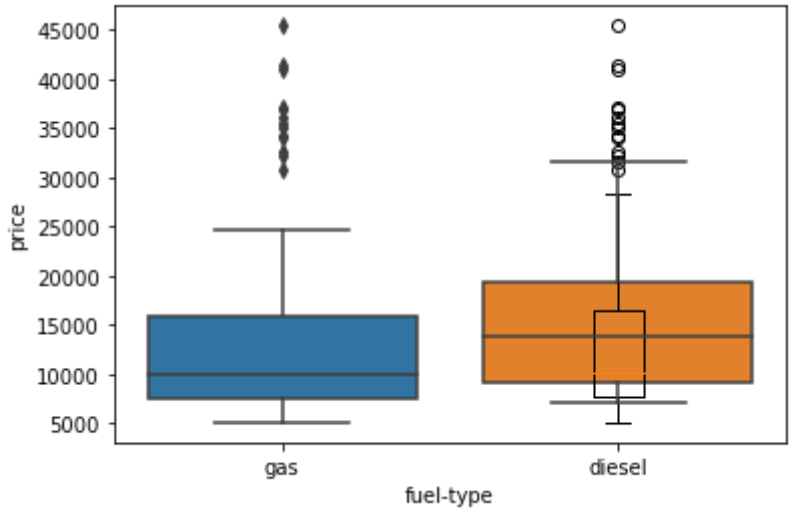


In [59]:

```
# examples of box plot
plt.boxplot(data['price'])

# by using seaborn
sns.boxplot(x ='fuel-type', y ='price', data = data)

plt.show()
```



In [51]:

```
# Grouping Data
test = data[['drive-wheels', 'make', 'price']]
data_grp = test.groupby(['drive-wheels', 'make'],
                        as_index = False).mean()

data_grp
```

Out[51]:

	drive-wheels	make	price
0	4wd	audi	17450.000000
1	4wd	subaru	9560.400000
2	4wd	toyota	8338.000000
3	fwd	audi	17941.000000
4	fwd	chevrolet	6007.000000
5	fwd	dodge	7875.444444
6	fwd	honda	8184.692308
7	fwd	mazda	8399.545455
8	fwd	mitsubishi	9239.769231
9	fwd	nissan	8812.333333
10	fwd	plymouth	7163.333333
11	fwd	renault	9595.000000
12	fwd	saab	15223.333333
13	fwd	subaru	7813.285714
14	fwd	toyota	8253.000000
15	fwd	volkswagen	10077.500000
16	rwd	alfa-romero	15498.333333
17	rwd	bmw	26118.750000
18	rwd	isuzu	8916.500000
19	rwd	jaguar	34600.000000
20	rwd	mazda	14784.000000
21	rwd	mercedes-benz	33647.000000
22	rwd	mercury	16503.000000
23	rwd	nissan	18432.333333

	drive-wheels	make	price
24	rwd	peugot	15489.090909
25	rwd	plymouth	12764.000000
26	rwd	porsche	31400.500000
27	rwd	toyota	11973.000000
28	rwd	volvo	18063.181818

In [52]:

```
# pivot method
data_pivot = data_grp.pivot(index = 'drive-wheels',
                            columns = 'make')
data_pivot
```

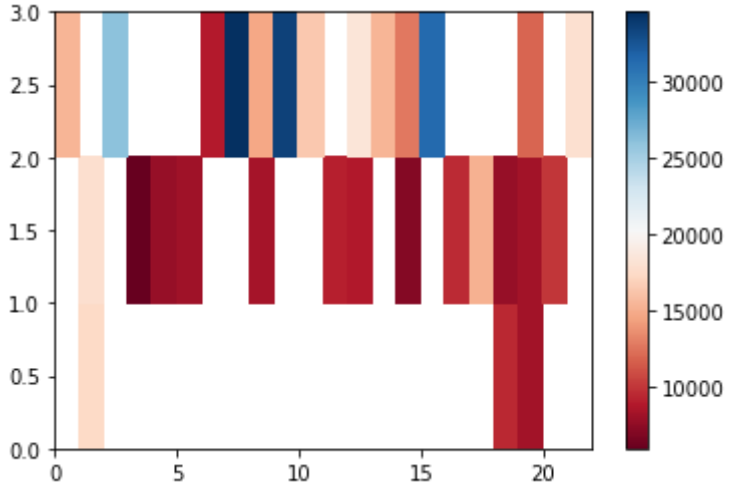
Out[52]:

	price											
make	**alfa-romero**	**audi**	**bmw**	**chevrolet**	**dodge**	**honda**	**isuzu**	**jaguar**	**mazda**	**mercedes-benz**	**...**	**nissan**
drive-wheels												
4wd	NaN	17450.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	N
fwd	NaN	17941.0	NaN	6007.0	7875.444444	8184.692308	NaN	NaN	8399.545455	NaN	...	8812.333
rwd	15498.333333	NaN	26118.75	NaN	NaN	NaN	8916.5	34600.0	14784.000000	33647.0	...	18432.333

3 rows × 22 columns

In [53]:

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



In [54]:

```
data_annova = data[['make', 'price']]
grouped_annova = data_annova.groupby(['make'])
annova_results_l = sp.stats.f_oneway(
                            grouped_annova.get_group('honda')['price'],
                            grouped_annova.get_group('subaru')['price']
                                   )
print(annova_results_l)
```

```
F_onewayResult(statistic=0.19744030127462606, pvalue=0.6609478240622193)
```

In [67]:

```
data_annova = data[['make', 'price']]
grouped_annova = data_annova.groupby(['make'])
annova_results_l = sp.stats.f_oneway(
                            grouped_annova.get_group('alfa-romero')['price'],
                            grouped_annova.get_group('audi')['price']
                                   )
print(annova_results_l)
```

```
F_onewayResult(statistic=1.1892083742577848, pvalue=0.3115982122629383)
```

In [68]:

```
data_annova = data[['make', 'price']]
grouped_annova = data_annova.groupby(['make'])
annova_results_l = sp.stats.f_oneway(
                            grouped_annova.get_group('audi')['price'],
                            grouped_annova.get_group('bmw')['price']
                                   )
print(annova_results_l)
```

```
F_onewayResult(statistic=4.2506287766018, pvalue=0.06158493578969085)
```

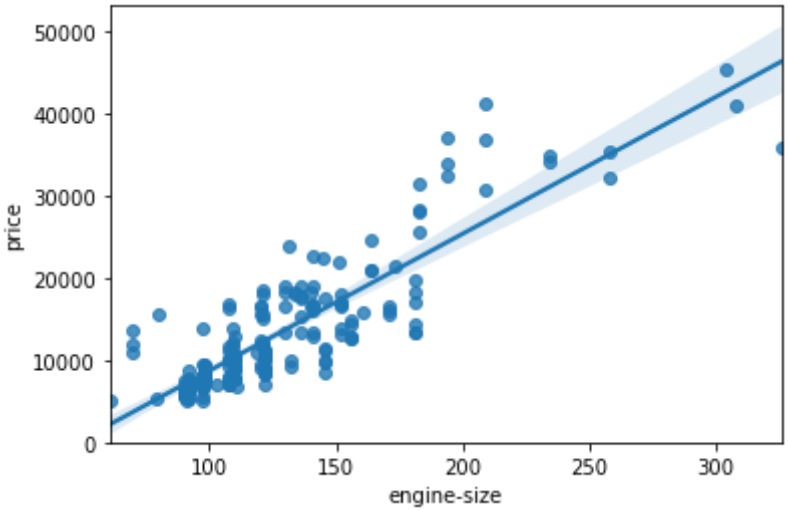
In [60]:

```
# strong corealtion between a categorical variable
# if annova test gives large f-test and small p-value

# Correlation- measures dependency, not causation
sns.regplot(x ='engine-size', y ='price', data = data)
plt.ylim(0, )
```

Out[60]:

```
(0.0, 53200.83987207094)
```



In [ ]:

```
from sklearn.linear_model import LinearRegression
lm=LinearRegression()
x=data[["engine-size"]]
y=data['price']
lm.fit(x,y)
yhat=lm.predict(x)
lm.intercept_
lm.coef_
print("intercept:",lm.intercept_)
print("coef:",lm.coef_)
```

```
intercept: -7963.338906281046
coef: [166.86001569]
```

```
hence we have price= -7963.34+166.87*engine-size
```

```
sns.regplot(x="engine-size",y="price",data=data)
plt.ylim(0,)
```

```
3]: (0.0, 53189.85252507509)
```