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]:
```
df = pd.read_csv(r"C:\Users\robin\Desktop\output.csv")
df.head()
```

Out[2]:

	Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Colum
0	3	?	alfa-romero	gas	std	two	convertible	rwd	fr
1	3	?	alfa-romero	gas	std	two	convertible	rwd	fr
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	fr
3	2	164	audi	gas	std	four	sedan	fwd	fr
4	2	164	audi	gas	std	four	sedan	4wd	fr

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	wh b
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 rows × 26 columns

In [7]:
```
data  = df
data.isna().any()
data.isnull().any()
```

Out[7]:
```
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 [8]:
```
data['city-mpg'] = 235/df['city-mpg']
data.rename(columns = {'city_mpg': "city-L/ 100km"}, inplace = True)

print(data.columns)

data.dtypes
```

```
Index(['symboling', 'normalized-losses', 'make', 'fuel-type', 'aspiratio
n',
       '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', 'strok
e',
       'compression-ratio', 'horsepower', 'peak-rpm', 'city-mpg',
       'highway-mpg', 'price'],
      dtype='object')
```

Out[8]:
```
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             float64
highway-mpg            int64
price                 object
dtype: object
```

In [9]:
```
data.price.unique()

data = data[data.price != '?']

data.dtypes
```

Out[9]:
```
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             float64
highway-mpg            int64
price                 object
dtype: object
```

In [10]:
```
data.price.unique()
```

Out[10]:
```
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', '998
8',
       '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 [11]:
```
data['length'] = data['length']/data['length'].max()
data['width'] = data['width']/data['width'].max()
data['height'] = data['height']/data['height'].max()

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()
```

```
C:\Users\robin\AppData\Local\Temp\ipykernel_15956\3639073924.py:1: Settin
gWithCopyWarning:
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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  data['length'] = data['length']/data['length'].max()
C:\Users\robin\AppData\Local\Temp\ipykernel_15956\3639073924.py:2: Settin
gWithCopyWarning:
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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  data['width'] = data['width']/data['width'].max()
C:\Users\robin\AppData\Local\Temp\ipykernel_15956\3639073924.py:3: Settin
gWithCopyWarning:
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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  data['height'] = data['height']/data['height'].max()
```

```
---------------------------------------------------------------------------
UFuncTypeError                            Traceback (most recent call last)
Cell In[11], line 5
      2 data['width'] = data['width']/data['width'].max()
      3 data['height'] = data['height']/data['height'].max()
----> 5 bins = np.linspace(min(data['price']), max (data['price']), 4)
      7 group_names = ['Low', 'Medium', 'High']
      8 data['price-binned'] = pd.cut(data['price'], bins,
      9                               labels = group_names,
     10                               include_lowest= True)

File <__array_function__ internals>:180, in linspace(*args, **kwargs)

File ~\anaconda3\lib\site-packages\numpy\core\function_base.py:127, in linspace(start, stop, num, endpoint, retstep, dtype, axis)
    123 div = (num - 1) if endpoint else num
    125 # Convert float/complex array scalars to float, gh-3504
    126 # and make sure one can use variables that have an __array_interface__, gh-6634
--> 127 start = asanyarray(start) * 1.0
    128 stop  = asanyarray(stop)  * 1.0
    130 dt = result_type(start, stop, float(num))

UFuncTypeError: ufunc 'multiply' did not contain a loop with signature matching types (dtype('<U5'), dtype('float64')) -> None
```

In [12]:
```
data['length'] = data['length']/data['length'].max()
data['width'] = data['width']/data['width'].max()
data['height'] = data['height']/data['height'].max()

# 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()
```

```
C:\Users\robin\AppData\Local\Temp\ipykernel_15956\923714955.py:1: Setting
WithCopyWarning:
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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  data['length'] = data['length']/data['length'].max()
C:\Users\robin\AppData\Local\Temp\ipykernel_15956\923714955.py:2: Setting
WithCopyWarning:
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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  data['width'] = data['width']/data['width'].max()
C:\Users\robin\AppData\Local\Temp\ipykernel_15956\923714955.py:3: Setting
WithCopyWarning:
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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  data['height'] = data['height']/data['height'].max()
```

```
---------------------------------------------------------------------------
UFuncTypeError                            Traceback (most recent call last)
Cell In[12], line 6
      3 data['height'] = data['height']/data['height'].max()
      5 # binning- grouping values
----> 6 bins = np.linspace(min(data['price']), max(data['price']), 4)
      7 group_names = ['Low', 'Medium', 'High']
      8 data['price-binned'] = pd.cut(data['price'], bins,
      9                                                       labels = group_names,
     10                                                       include_lowest = True)

File <__array_function__ internals>:180, in linspace(*args, **kwargs)

File ~\anaconda3\lib\site-packages\numpy\core\function_base.py:127, in linspace(start, stop, num, endpoint, retstep, dtype, axis)
    123 div = (num - 1) if endpoint else num
    125 # Convert float/complex array scalars to float, gh-3504
    126 # and make sure one can use variables that have an __array_interface__, gh-6634
--> 127 start = asanyarray(start) * 1.0
    128 stop  = asanyarray(stop)  * 1.0
    130 dt = result_type(start, stop, float(num))

UFuncTypeError: ufunc 'multiply' did not contain a loop with signature matching types (dtype('<U5'), dtype('float64')) -> None
```

In [14]:

```
pip install upgrade pip setuptools wheel
pip install bertopic --no-cache-dir
pip uninstall hdbscan -y
pip install hdbscan --no-cache-dir --no-binary :all: --no-build-isolation
```

```
  Cell In[14], line 1
    pip install upgrade pip setuptools wheel
        ^
SyntaxError: invalid syntax
```

In [15]:
```
# categorical to numerical variables
pd.get_dummies(data['fuel-type']).head()

# descriptive analysis
# NaN are skipped
data.describe()
```

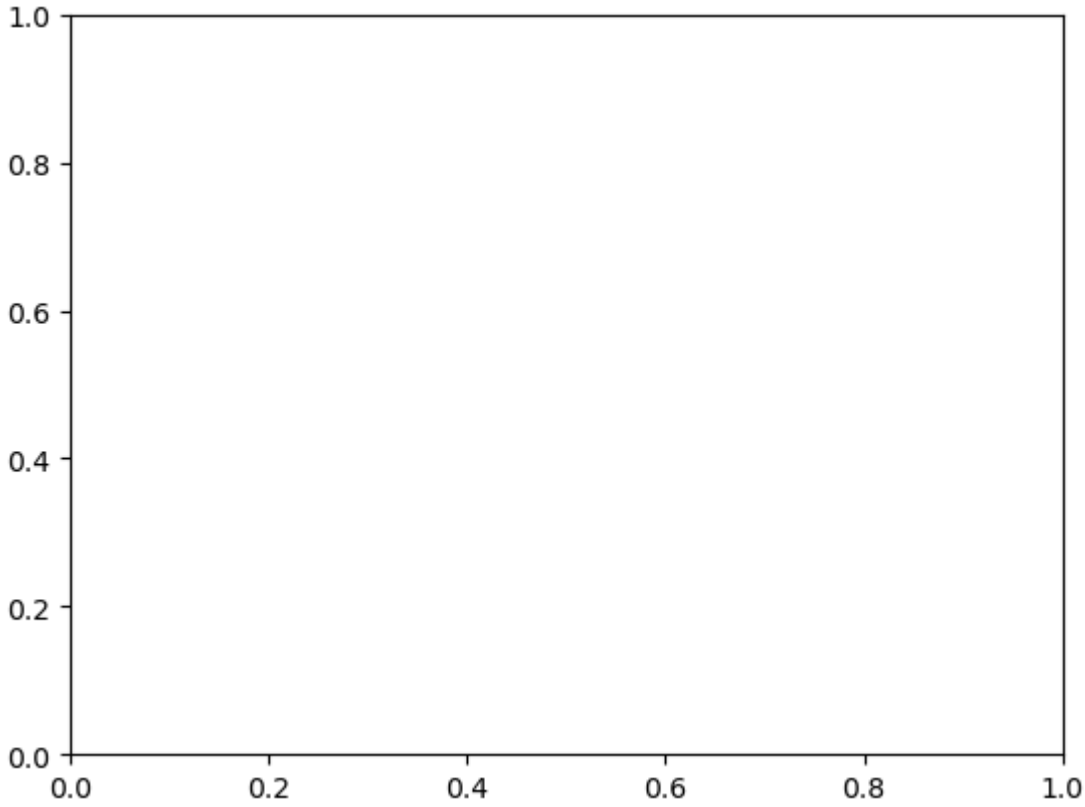
Out[15]:

	symboling	wheel-base	length	width	height	curb-weight	engine-size
count	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000
mean	0.840796	98.797015	0.837102	0.915126	0.899108	2555.666667	126.875622
std	1.254802	6.066366	0.059213	0.029187	0.040933	517.296727	41.546834
min	-2.000000	86.600000	0.678039	0.837500	0.799331	1488.000000	61.000000
25%	0.000000	94.500000	0.801538	0.890278	0.869565	2169.000000	98.000000
50%	1.000000	97.000000	0.832292	0.909722	0.904682	2414.000000	120.000000
75%	2.000000	102.400000	0.881788	0.925000	0.928094	2926.000000	141.000000
max	3.000000	120.900000	1.000000	1.000000	1.000000	4066.000000	326.000000

In [16]:
```
# 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()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[16], line 2
      1 # examples of box plot
----> 2 plt.boxplot(data['price'])
      4 # by using seaborn
      5 sns.boxplot(x ='drive-wheels', y ='price', data = data)

File ~\anaconda3\lib\site-packages\matplotlib\pyplot.py:2456, in boxplot(x, notch, sym, vert, whis, positions, widths, patch_artist, bootstrap, usermedians, conf_intervals, meanline, showmeans, showcaps, showbox, showfliers, boxprops, labels, flierprops, medianprops, meanprops, capprops, whiskerprops, manage_ticks, autorange, zorder, capwidths, data)
   2446 @_copy_docstring_and_deprecators(Axes.boxplot)
   2447 def boxplot(
   2448         x, notch=None, sym=None, vert=None, whis=None,
   (...)
   2454         whiskerprops=None, manage_ticks=True, autorange=False,
   2455         zorder=None, capwidths=None, *, data=None):
-> 2456     return gca().boxplot(
   2457         x, notch=notch, sym=sym, vert=vert, whis=whis,
   2458         positions=positions, widths=widths, patch_artist=patch_artist,
   2459         bootstrap=bootstrap, usermedians=usermedians,
   2460         conf_intervals=conf_intervals, meanline=meanline,
   2461         showmeans=showmeans, showcaps=showcaps, showbox=showbox,
   2462         showfliers=showfliers, boxprops=boxprops, labels=labels,
   2463         flierprops=flierprops, medianprops=medianprops,
   2464         meanprops=meanprops, capprops=capprops,
   2465         whiskerprops=whiskerprops, manage_ticks=manage_ticks,
   2466         autorange=autorange, zorder=zorder, capwidths=capwidths,
   2467         **({"data": data} if data is not None else {}))

File ~\anaconda3\lib\site-packages\matplotlib\__init__.py:1442, in _preprocess_data.<locals>.inner(ax, data, *args, **kwargs)
   1439 @functools.wraps(func)
   1440 def inner(ax, *args, data=None, **kwargs):
   1441     if data is None:
-> 1442         return func(ax, *map(sanitize_sequence, args), **kwargs)
   1444     bound = new_sig.bind(ax, *args, **kwargs)
   1445     auto_label = (bound.arguments.get(label_namer)
   1446                   or bound.kwargs.get(label_namer))

File ~\anaconda3\lib\site-packages\matplotlib\axes\_axes.py:3914, in Axes.boxplot(self, x, notch, sym, vert, whis, positions, widths, patch_artist, bootstrap, usermedians, conf_intervals, meanline, showmeans, showcaps, showbox, showfliers, boxprops, labels, flierprops, medianprops, meanprops, capprops, whiskerprops, manage_ticks, autorange, zorder, capwidths)
   3911 if bootstrap is None:
   3912     bootstrap = mpl.rcParams['boxplot.bootstrap']
-> 3914 bxpstats = cbook.boxplot_stats(x, whis=whis, bootstrap=bootstrap,
   3915                                 labels=labels, autorange=autorange)
   3916 if notch is None:
   3917     notch = mpl.rcParams['boxplot.notch']

File ~\anaconda3\lib\site-packages\matplotlib\cbook\__init__.py:1232, in boxplot_stats(X, whis, bootstrap, labels, autorange)
   1229 x = np.asarray(x)
```

```
   1231 # arithmetic mean
-> 1232 stats['mean'] = np.mean(x)
   1234 # medians and quartiles
   1235 q1, med, q3 = np.percentile(x, [25, 50, 75])

File <__array_function__ internals>:180, in mean(*args, **kwargs)

File ~\anaconda3\lib\site-packages\numpy\core\fromnumeric.py:3432, in mea
n(a, axis, dtype, out, keepdims, where)
   3429     else:
   3430         return mean(axis=axis, dtype=dtype, out=out, **kwargs)
-> 3432 return _methods._mean(a, axis=axis, dtype=dtype,
   3433                      out=out, **kwargs)

File ~\anaconda3\lib\site-packages\numpy\core\_methods.py:192, in _mean
(a, axis, dtype, out, keepdims, where)
    190         ret = ret.dtype.type(ret / rcount)
    191 else:
--> 192     ret = ret / rcount
    194 return ret

TypeError: ufunc 'divide' not supported for the input types, and the inpu
ts could not be safely coerced to any supported types according to the ca
sting rule ''safe''
```

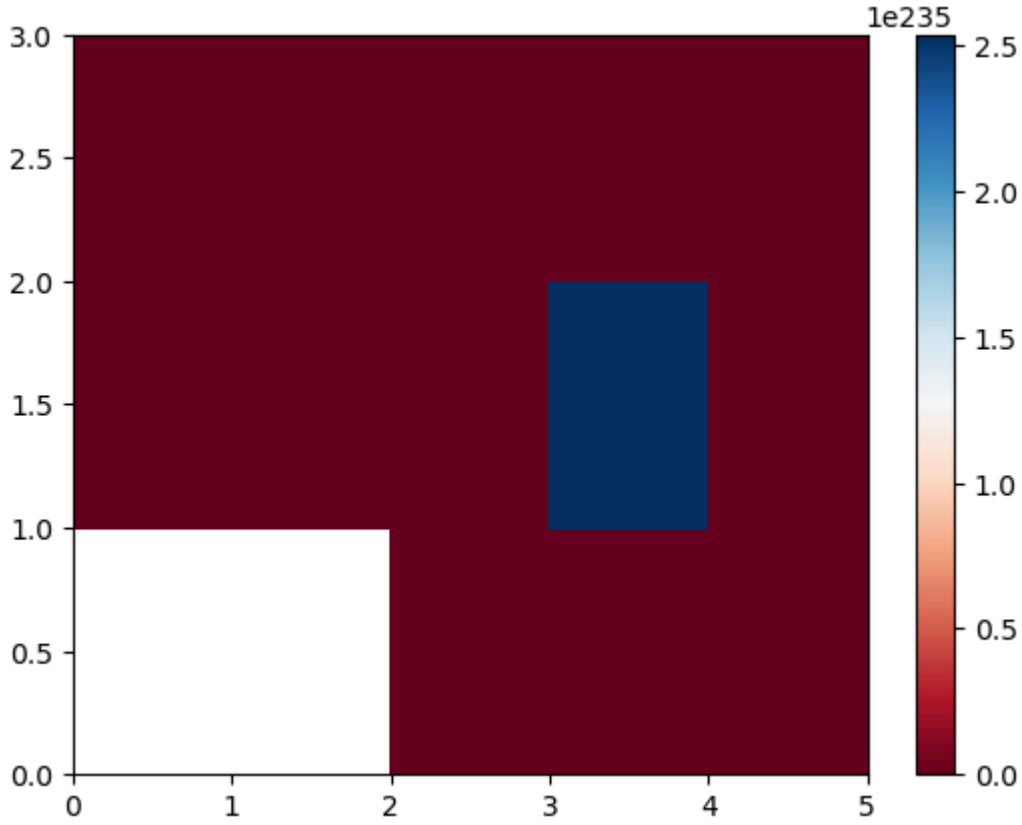


In [17]:
```
# Grouping Data
test = data[['drive-wheels', 'body-style', 'price']]
data_grp = test.groupby(['drive-wheels', 'body-style'],
                        as_index = False).mean()

data_grp
```

Out[17]:

	drive-wheels	body-style	price
0	4wd	hatchback	7.603000e+03
1	4wd	sedan	5.816974e+12
2	4wd	wagon	2.003279e+16
3	fwd	convertible	1.159500e+04
4	fwd	hardtop	8.249000e+03
5	fwd	hatchback	1.051353e+204
6	fwd	sedan	2.536391e+235
7	fwd	wagon	1.576741e+50
8	rwd	convertible	2.699033e+23
9	rwd	hardtop	4.025208e+31
10	rwd	hatchback	9.166728e+84
11	rwd	sedan	4.563936e+174
12	rwd	wagon	3.138680e+43

In [18]:
```
# pivot method
data_pivot = data_grp.pivot(index = 'drive-wheels',
                            columns = 'body-style')
data_pivot

# heatmap for visualizing data
plt.pcolor(data_pivot, cmap ='RdBu')
plt.colorbar()
plt.show()
```

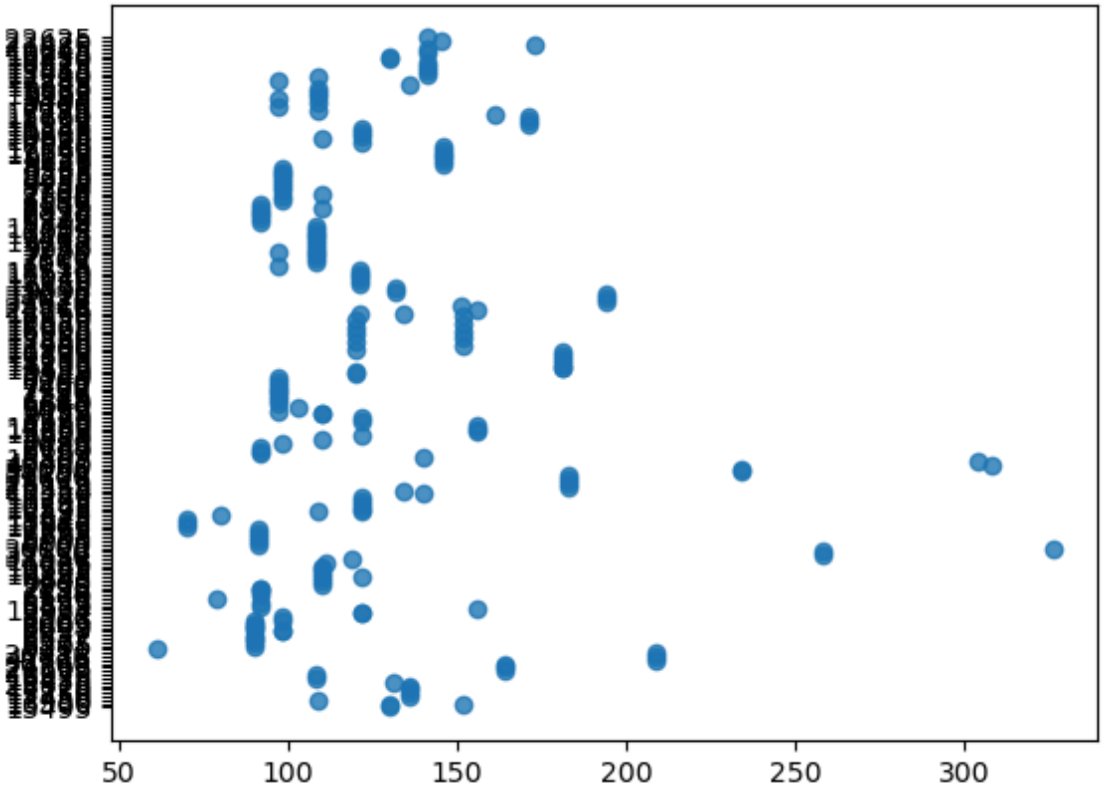


In [20]:
```
# Analysis of Variance- ANOVA
# returns f-test and p-value
# f-test = variance between sample group means divided by
# variation within sample group
# p-value = confidence degree
data_annova = data[['make', 'price']]
grouped_annova = data_annova.groupby(['make'])
annova_results_l = sp.stats.f_oneway(
                            grouped_annova.get_group('honda')['price'],
                            grouped_annova.get_group('subaru')['price']
                                )
print(annova_results_l)

# 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, )
```

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

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[20], line 18
     12 print(annova_results_l)
     14 # strong corealtion between a categorical variable
     15 # if annova test gives large f-test and small p-value
     16
     17 # Correlation- measures dependency, not causation
---> 18 sns.regplot(x ='engine-size', y ='price', data = data)
     19 plt.ylim(0, )

File ~\anaconda3\lib\site-packages\seaborn\regression.py:759, in regplot(data, x, y, x_estimator, x_bins, x_ci, scatter, fit_reg, ci, n_boot, units, seed, order, logistic, lowess, robust, logx, x_partial, y_partial, truncate, dropna, x_jitter, y_jitter, label, color, marker, scatter_kws, line_kws, ax)
    757 scatter_kws["marker"] = marker
    758 line_kws = {} if line_kws is None else copy.copy(line_kws)
--> 759 plotter.plot(ax, scatter_kws, line_kws)
    760 return ax

File ~\anaconda3\lib\site-packages\seaborn\regression.py:368, in _RegressionPlotter.plot(self, ax, scatter_kws, line_kws)
    365     self.scatterplot(ax, scatter_kws)
    367 if self.fit_reg:
--> 368     self.lineplot(ax, line_kws)
    370 # Label the axes
    371 if hasattr(self.x, "name"):

File ~\anaconda3\lib\site-packages\seaborn\regression.py:413, in _RegressionPlotter.lineplot(self, ax, kws)
    411 """Draw the model."""
    412 # Fit the regression model
--> 413 grid, yhat, err_bands = self.fit_regression(ax)
    414 edges = grid[0], grid[-1]
    416 # Get set default aesthetics

File ~\anaconda3\lib\site-packages\seaborn\regression.py:219, in _RegressionPlotter.fit_regression(self, ax, x_range, grid)
    217     yhat, yhat_boots = self.fit_logx(grid)
    218 else:
--> 219     yhat, yhat_boots = self.fit_fast(grid)
    221 # Compute the confidence interval at each grid point
    222 if ci is None:

File ~\anaconda3\lib\site-packages\seaborn\regression.py:236, in _RegressionPlotter.fit_fast(self, grid)
    234 X, y = np.c_[np.ones(len(self.x)), self.x], self.y
    235 grid = np.c_[np.ones(len(grid)), grid]
--> 236 yhat = grid.dot(reg_func(X, y))
    237 if self.ci is None:
    238     return yhat, None

File ~\anaconda3\lib\site-packages\seaborn\regression.py:232, in _RegressionPlotter.fit_fast.<locals>.reg_func(_x, _y)
    231 def reg_func(_x, _y):
--> 232     return np.linalg.pinv(_x).dot(_y)
```

```
TypeError: can't multiply sequence by non-int of type 'float'
```



In [ ]: