1. Import Data

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
path='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-SkillsNetwork/labs/Data%20files/automobileEDA.csv'
df = pd.read_csv(path)
df.head()
```

Out[2]:

	symboling	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	...	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price	city-L/100km	horsepower-binned	diesel	gas
0	3	122	alfa-romero	std	two	convertible	rwd	front	88.6	0.811148	...	9.0	111.0	5000.0	21	27	13495.0	11.190476	Medium	0	1
1	3	122	alfa-romero	std	two	convertible	rwd	front	88.6	0.811148	...	9.0	111.0	5000.0	21	27	16500.0	11.190476	Medium	0	1
2	1	122	alfa-romero	std	two	hatchback	rwd	front	94.5	0.822681	...	9.0	154.0	5000.0	19	26	16500.0	12.368421	Medium	0	1
3	2	164	audi	std	four	sedan	fwd	front	99.8	0.848630	...	10.0	102.0	5500.0	24	30	13950.0	9.791667	Medium	0	1
4	2	164	audi	std	four	sedan	4wd	front	99.4	0.848630	...	8.0	115.0	5500.0	18	22	17450.0	13.055556	Medium	0	1

5 rows × 29 columns

1. Analyzing Individual Feature Patterns Using Visualization

In [3]:
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [4]:
```
print(df.dtypes)
```

```
symboling              int64
normalized-losses      int64
make                  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                 float64
stroke               float64
compression-ratio    float64
horsepower           float64
peak-rpm             float64
city-mpg               int64
highway-mpg            int64
price                float64
city-L/100km         float64
horsepower-binned     object
diesel                 int64
gas                    int64
dtype: object
```

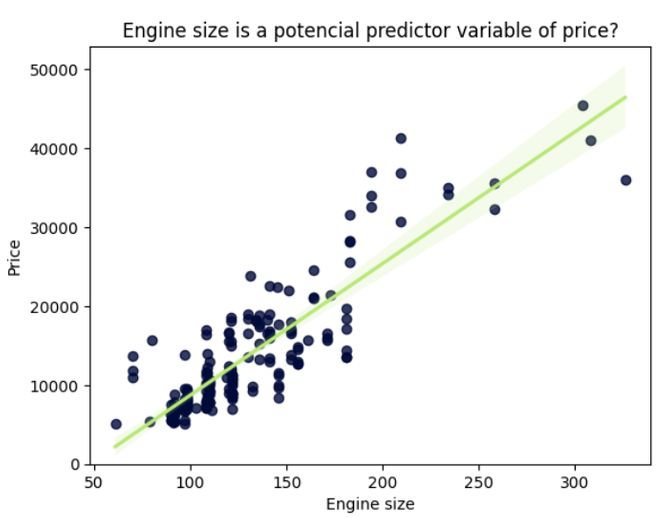
In [5]:
```
df[['bore','stroke','compression-ratio','horsepower']].corr()
```

Out[5]:

	bore	stroke	compression-ratio	horsepower
bore	1.000000	-0.055390	0.001263	0.566936
stroke	-0.055390	1.000000	0.187923	0.098462
compression-ratio	0.001263	0.187923	1.000000	-0.214514
horsepower	0.566936	0.098462	-0.214514	1.000000

In [6]:
```
sns.regplot(x="engine-size", y="price", data=df, scatter_kws={"color": "#000a38"}, line_kws={"color": "#b9e878"})
plt.xlabel("Engine size")
plt.ylabel("Price")
plt.title("Engine size is a potencial predictor variable of price?")
plt.ylim(0,)
```

Out[6]:
```
(0.0, 52890.37670598886)
```



In [7]:
```
df[["engine-size", "price"]].corr()
```

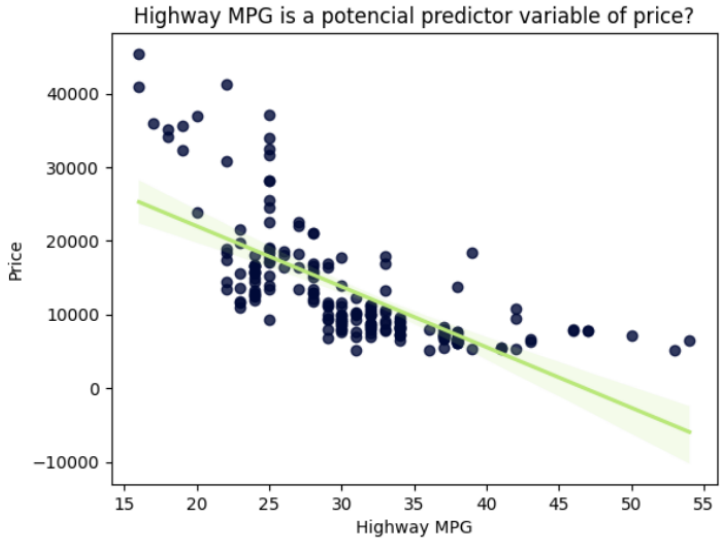
Out[7]:

	engine-size	price

engine-size	1.000000	0.872335
price	0.872335	1.000000

In [8]:
```
sns.regplot(x="highway-mpg", y="price", data=df,scatter_kws={"color": "#000a38"}, line_kws={"color": "#b9e878"})
plt.xlabel("Highway MPG")
plt.ylabel("Price")
plt.title("Highway MPG is a potencial predictor variable of price?")
```

Out[8]:
```
Text(0.5, 1.0, 'Highway MPG is a potencial predictor variable of price?')
```



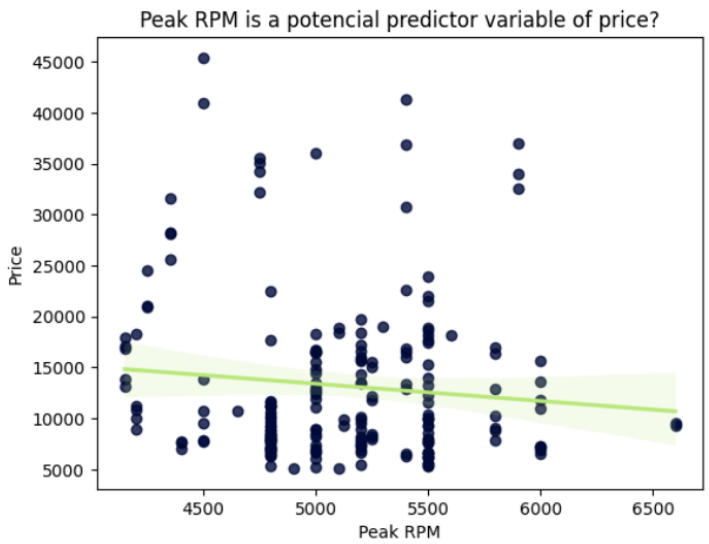
In [9]:
```
df[['highway-mpg', 'price']].corr()
```

Out[9]:

	highway-mpg	price
highway-mpg	1.000000	-0.704692
price	-0.704692	1.000000

In [10]:
```
sns.regplot(x="peak-rpm", y="price", data=df, scatter_kws={"color": "#000a38"}, line_kws={"color": "#b9e878"})
plt.xlabel("Peak RPM")
plt.ylabel("Price")
plt.title("Peak RPM is a potencial predictor variable of price?")
```

Out[10]:
```
Text(0.5, 1.0, 'Peak RPM is a potencial predictor variable of price?')
```



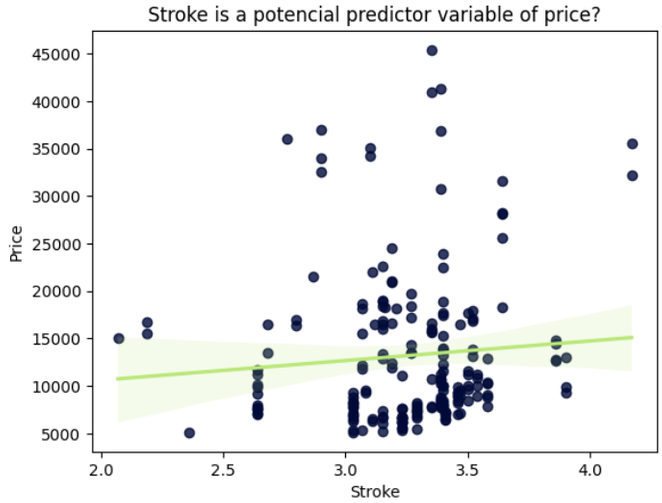
In [11]:
```
df[['peak-rpm','price']].corr()
```

Out[11]:

	peak-rpm	price
peak-rpm	1.000000	-0.101616
price	-0.101616	1.000000

In [12]:
```
sns.regplot(x="stroke", y="price", data=df, scatter_kws={"color": "#000a38"}, line_kws={"color": "#b9e878"})
plt.xlabel("Stroke")
plt.ylabel("Price")
plt.title("Stroke is a potencial predictor variable of price?")
```

Out[12]:
```
Text(0.5, 1.0, 'Stroke is a potencial predictor variable of price?')
```



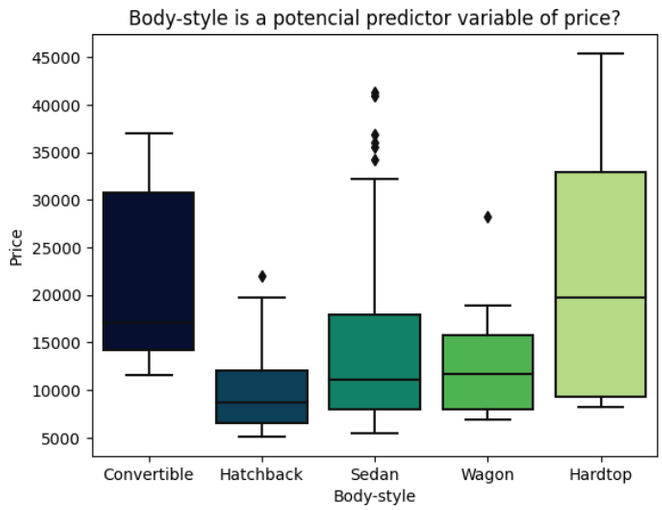
In [13]:
```
df[['stroke','price']].corr()
```

Out[13]:

	stroke	price
stroke	1.00000	0.08231
price	0.08231	1.00000

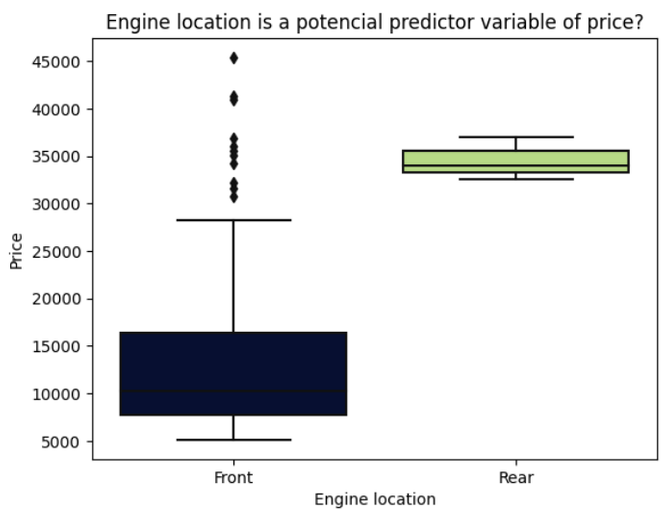
In [14]:
```
my_pal = {"convertible": "#000a38", "hatchback": "#004364", "sedan":"#00936e", "wagon":"#3ec443", "hardtop":"#b9e878"}
ax=sns.boxplot(x="body-style", y="price", data=df, palette=my_pal)
plt.xlabel("Body-style")
plt.ylabel("Price")
plt.title("Body-style is a potencial predictor variable of price?")
ax.set_xticklabels(["Convertible", "Hatchback", "Sedan", "Wagon","Hardtop"])
```

Out[14]:
```
[Text(0, 0, 'Convertible'),
 Text(1, 0, 'Hatchback'),
 Text(2, 0, 'Sedan'),
 Text(3, 0, 'Wagon'),
 Text(4, 0, 'Hardtop')]
```



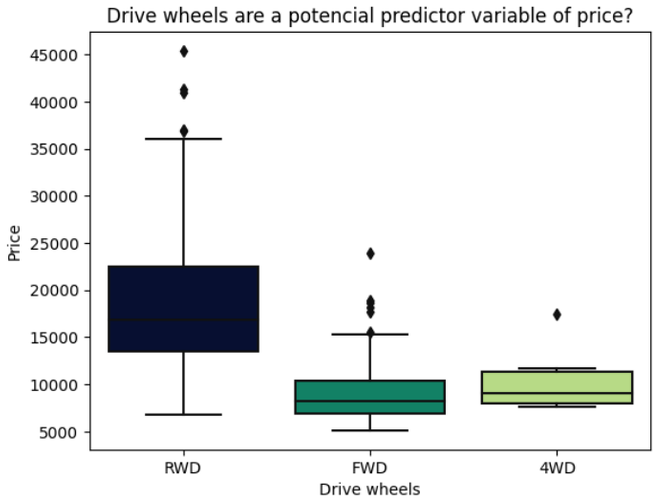
In [15]:
```
my_pal = {"front": "#000a38", "rear": "#b9e878"}
ax=sns.boxplot(x="engine-location", y="price", data=df, palette=my_pal)
plt.xlabel("Engine location")
plt.ylabel("Price")
plt.title("Engine location is a potencial predictor variable of price?")
ax.set_xticklabels(["Front", "Rear"])
```

Out[15]:
```
[Text(0, 0, 'Front'), Text(1, 0, 'Rear')]
```



In [16]:
```
my_pal = {"rwd": "#000a38","fwd": "#00936e", "4wd": "#b9e878",}
ax=sns.boxplot(x="drive-wheels", y="price", data=df, palette=my_pal)
plt.xlabel("Drive wheels")
plt.ylabel("Price")
plt.title("Drive wheels are a potencial predictor variable of price?")
ax.set_xticklabels(["RWD", "FWD", "4WD"])
```

Out[16]: [Text(0, 0, 'RWD'), Text(1, 0, 'FWD'), Text(2, 0, '4WD')]



1. Descriptive Statistical Analysis

In [17]:
```
df.describe()
```

Out[17]:

	symboling	normalized-losses	wheel-base	length	width	height	curb-weight	engine-size	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg
count	201.000000	201.00000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	197.000000	201.000000	201.000000	201.000000	201.000000	201.000000
mean	0.840796	122.00000	98.797015	0.837102	0.915126	53.766667	2555.666667	126.875622	3.330692	3.256904	10.164279	103.405534	5117.665368	25.179104	30.686567
std	1.254802	31.99625	6.066366	0.059213	0.029187	2.447822	517.296727	41.546834	0.268072	0.319256	4.004965	37.365700	478.113805	6.423220	6.815150
min	-2.000000	65.00000	86.600000	0.678039	0.837500	47.800000	1488.000000	61.000000	2.540000	2.070000	7.000000	48.000000	4150.000000	13.000000	16.000000
25%	0.000000	101.00000	94.500000	0.801538	0.890278	52.000000	2169.000000	98.000000	3.150000	3.110000	8.600000	70.000000	4800.000000	19.000000	25.000000
50%	1.000000	122.00000	97.000000	0.832292	0.909722	54.100000	2414.000000	120.000000	3.310000	3.290000	9.000000	95.000000	5125.369458	24.000000	30.000000
75%	2.000000	137.00000	102.400000	0.881788	0.925000	55.500000	2926.000000	141.000000	3.580000	3.410000	9.400000	116.000000	5500.000000	30.000000	34.000000
max	3.000000	256.00000	120.900000	1.000000	1.000000	59.800000	4066.000000	326.000000	3.940000	4.170000	23.000000	262.000000	6600.000000	49.000000	54.000000

In [18]:
```
df.describe(include=['object'])
```

Out[18]:

	make	aspiration	num-of-doors	body-style	drive-wheels	engine-location	engine-type	num-of-cylinders	fuel-system	horsepower-binned
count	201	201	201	201	201	201	201	201	201	200
unique	22	2	2	5	3	2	6	7	8	3
top	toyota	std	four	sedan	fwd	front	ohc	four	mpfi	Low
freq	32	165	115	94	118	198	145	157	92	115

In [19]:
```
df['drive-wheels'].value_counts()
```

Out[19]:
```
fwd    118
rwd     75
4wd      8
Name: drive-wheels, dtype: int64
```

In [20]:
```
df['drive-wheels'].value_counts().to_frame()
```

Out[20]:

	drive-wheels
fwd	118
rwd	75
4wd	8

In [21]:
```
drive_wheels_counts = df['drive-wheels'].value_counts().to_frame()
drive_wheels_counts.rename(columns={'drive-wheels': 'value_counts'}, inplace=True)
drive_wheels_counts
```

Out[21]:

	value_counts
fwd	118
rwd	75
4wd	8

In [22]:
```
drive_wheels_counts.index.name = 'drive-wheels'
drive_wheels_counts
```

Out[22]:

drive-wheels	value_counts
fwd	118
rwd	75
4wd	8

In [23]:
```
engine_loc_counts = df['engine-location'].value_counts().to_frame()
engine_loc_counts.rename(columns={'engine-location': 'value_counts'}, inplace=True)
engine_loc_counts.index.name = 'engine-location'
engine_loc_counts.head(10)
```

Out[23]:

engine-location	value_counts
front	198
rear	3

1. Basics of Grouping

In [24]:
```
df['drive-wheels'].unique()
```

Out[24]:
```
array(['rwd', 'fwd', '4wd'], dtype=object)
```

In [25]:
```
df_group_one = df[['drive-wheels','body-style','price']]
df_group_one
```

Out[25]:

	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

In [26]:
```
df_group_one = df_group_one.groupby(['drive-wheels'],as_index=False).mean()
df_group_one
```

```
<ipython-input-26-f835cd4f37e0>:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future version, numeric_only will defau
lt to False. Either specify numeric_only or select only columns which should be valid for the function.
  df_group_one = df_group_one.groupby(['drive-wheels'],as_index=False).mean()
```

Out[26]:

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

In [27]:
```
df_gptest = df[['drive-wheels','body-style','price']]
grouped_test1 = df_gptest.groupby(['drive-wheels','body-style'],as_index=False).mean()
grouped_test1
```

Out[27]:

	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

In [28]:
```
grouped_pivot = grouped_test1.pivot(index='drive-wheels',columns='body-style')
grouped_pivot = grouped_pivot.fillna(0) #fill missing values with 0
grouped_pivot
```

Out[28]:

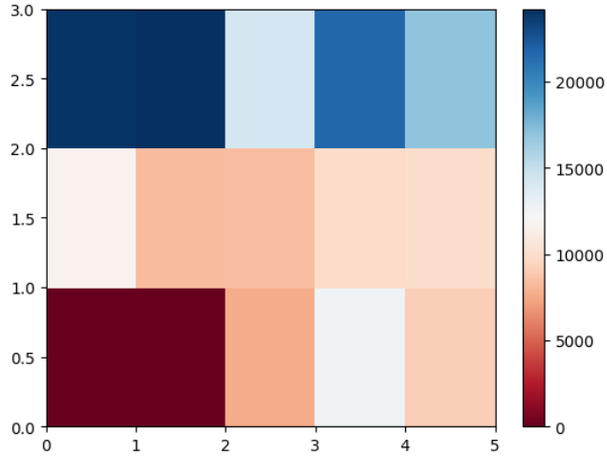
	price				
body-style	**convertible**	**hardtop**	**hatchback**	**sedan**	**wagon**
drive-wheels					
4wd	0.0	0.000000	7603.000000	12647.333333	9095.750000
fwd	11595.0	8249.000000	8396.387755	9811.800000	9997.333333
rwd	23949.6	24202.714286	14337.777778	21711.833333	16994.222222

In [29]:
```
df_gptest2 = df[['body-style','price']]
grouped_test2 = df_gptest2.groupby(['body-style'],as_index=False).mean()
grouped_test2
```

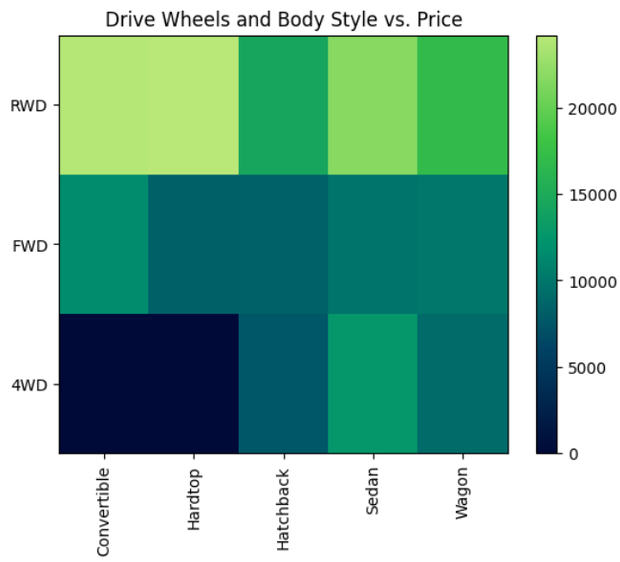
Out[29]:

	body-style	price
0	convertible	21890.500000
1	hardtop	22208.500000
2	hatchback	9957.441176
3	sedan	14459.755319
4	wagon	12371.960000

In [30]:
```
plt.pcolor(grouped_pivot, cmap='RdBu')
plt.colorbar()
plt.show()
```



In [31]:
```
from matplotlib.colors import LinearSegmentedColormap

def create_custom_colormap(colors, positions=None):
    if positions is None:
        positions = np.linspace(0, 1, len(colors))
    return LinearSegmentedColormap.from_list("", list(zip(positions, colors)))

colors = ["#000a38","#004364","#00936e","#3ec443","#b9e878"]

custom_colormap = create_custom_colormap(colors)

fig, ax = plt.subplots()
im = ax.pcolor(grouped_pivot, cmap=custom_colormap)

#label names
row_labels = grouped_pivot.columns.levels[1]
col_labels = grouped_pivot.index

#move ticks and labels to the center
ax.set_xticks(np.arange(grouped_pivot.shape[1]) + 0.5, minor=False)
ax.set_yticks(np.arange(grouped_pivot.shape[0]) + 0.5, minor=False)

#insert labels
ax.set_yticklabels(["4WD", "FWD", "RWD"])
ax.set_xticklabels(["Convertible", "Hardtop", "Hatchback", "Sedan", "Wagon"])
plt.title("Drive Wheels and Body Style vs. Price")
#rotate label if too long
plt.xticks(rotation=90)

fig.colorbar(im)
plt.show()
```



1. Correlation and Causation

In [32]:
```
from scipy import stats
```

In [33]:
```
pearson_coef, p_value = stats.pearsonr(df['wheel-base'], df['price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p_value)
```

```
The Pearson Correlation Coefficient is 0.584641822265508  with a P-value of P = 8.076488270732885e-20
```

In [34]:
```
pearson_coef, p_value = stats.pearsonr(df['horsepower'], df['price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p_value)
```

```
The Pearson Correlation Coefficient is 0.8095745670036559  with a P-value of P = 6.369057428259557e-48
```

In [35]:
```
pearson_coef, p_value = stats.pearsonr(df['length'], df['price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P = ", p_value)
```

```
The Pearson Correlation Coefficient is 0.6906283804483638  with a P-value of P =  8.016477466159723e-30
```

In [36]:
```
pearson_coef, p_value = stats.pearsonr(df['width'], df['price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p_value )
```

```
The Pearson Correlation Coefficient is 0.7512653440522673  with a P-value of P = 9.20033551048206e-38
```

In [37]:
```
pearson_coef, p_value = stats.pearsonr(df['curb-weight'], df['price'])
print( "The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P = ", p_value)
```

```
The Pearson Correlation Coefficient is 0.8344145257702843  with a P-value of P =  2.189577238893965e-53
```

In [38]:
```
pearson_coef, p_value = stats.pearsonr(df['engine-size'], df['price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p_value)
```

```
The Pearson Correlation Coefficient is 0.8723351674455185  with a P-value of P = 9.265491622198793e-64
```

In [39]:
```
pearson_coef, p_value = stats.pearsonr(df['bore'], df['price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p_value)
```

```
The Pearson Correlation Coefficient is 0.5431553832626602  with a P-value of P = 8.049189483935315e-17
```

In [40]:
```
pearson_coef, p_value = stats.pearsonr(df['city-mpg'], df['price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P = ", p_value)
```

```
The Pearson Correlation Coefficient is -0.6865710067844678  with a P-value of P =  2.3211320655675098e-29
```

In [41]:
```
pearson_coef, p_value = stats.pearsonr(df['highway-mpg'], df['price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P = ", p_value)
```

```
The Pearson Correlation Coefficient is -0.704692265058953  with a P-value of P =  1.749547114447557e-31
```

1. ANOVA

In [42]:
```
grouped_test3=df_gptest[['drive-wheels', 'price']].groupby(['drive-wheels'])
grouped_test3.head()
```

Out[42]:

	drive-wheels	price
0	rwd	13495.0
1	rwd	16500.0
2	rwd	16500.0
3	fwd	13950.0
4	4wd	17450.0
5	fwd	15250.0
6	fwd	17710.0
7	fwd	18920.0
8	fwd	23875.0
9	rwd	16430.0
10	rwd	16925.0
136	4wd	7603.0
140	4wd	9233.0
141	4wd	11259.0
144	4wd	8013.0

In [43]:
```
df_gptest
```

Out[43]:

	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

In [44]:
```
grouped_test3.get_group('4wd')['price']
```

Out[44]:
```
4       17450.0
136      7603.0
140      9233.0
141     11259.0
144      8013.0
145     11694.0
150      7898.0
151      8778.0
Name: price, dtype: float64
```

In [45]:
```
f_val, p_val = stats.f_oneway(grouped_test3.get_group('fwd')['price'], grouped_test3.get_group('rwd')['price'], grouped_test3.get_group('4wd')['price'])

print( "ANOVA results: F=", f_val, ", P =", p_val)
```

```
ANOVA results: F= 67.95406500780399 , P = 3.3945443577151245e-23
```

In [46]:
```
f_val, p_val = stats.f_oneway(grouped_test3.get_group('fwd')['price'], grouped_test3.get_group('rwd')['price'])

print( "ANOVA results: F=", f_val, ", P =", p_val )
```

```
ANOVA results: F= 130.5533160959111 , P = 2.2355306355677845e-23
```

In [47]:
```
f_val, p_val = stats.f_oneway(grouped_test3.get_group('4wd')['price'], grouped_test3.get_group('rwd')['price'])

print( "ANOVA results: F=", f_val, ", P =", p_val)
```

```
ANOVA results: F= 8.580681368924756 , P = 0.004411492211225333
```

In [48]:
```
f_val, p_val = stats.f_oneway(grouped_test3.get_group('4wd')['price'], grouped_test3.get_group('fwd')['price'])

print("ANOVA results: F=", f_val, ", P =", p_val)
```

```
ANOVA results: F= 0.665465750252303 , P = 0.41620116697845666
```