Exploratory Data Analysis

Objective:

Descriptive Statistics [describe, scatterplot, boxplot, heatmap] , GroupBy, pivot table, ANOVA, Correlation, Correlation-statistics [correlation coefficient, P-value] , Chi-squarred test etc

Import Dataset

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

In [3]:
```
import requests
def download (url, filename):
    response = requests.get(url)
    if response.status_code ==200:
        with open(filename, "wb") as f:
            f.write(response.content)
file_path = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA010
download(file_path, "cars.csv")
file_name = "cars.csv"
```

In [5]:
```
df = pd.read_csv(file_name, header = 0)
```

In [7]:
```
df.head(3)
```

Out[7]:

	symboling	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	...	compression-ratio	horsepowe
0	3	122	alfa-romero	std	two	convertible	rwd	front	88.6	0.811148	...	9.0	111.(
1	3	122	alfa-romero	std	two	convertible	rwd	front	88.6	0.811148	...	9.0	111.(
2	1	122	alfa-romero	std	two	hatchback	rwd	front	94.5	0.822681	...	9.0	154.(

3 rows × 29 columns

Analyzing Individual Feature Patterns Using Visualization

What is the data type of column "peak-rpm"?

In [11]:
```
df.columns
df["peak-rpm"].dtypes
```

Out[11]:
```
dtype('float64')
```

Find the correlation between the following columns: bore, stroke, compression-ratio, and horsepower.

In [13]:
```
df[["bore", "stroke", "compression-ratio","horsepower"]].corr()
```

Out[13]:

	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

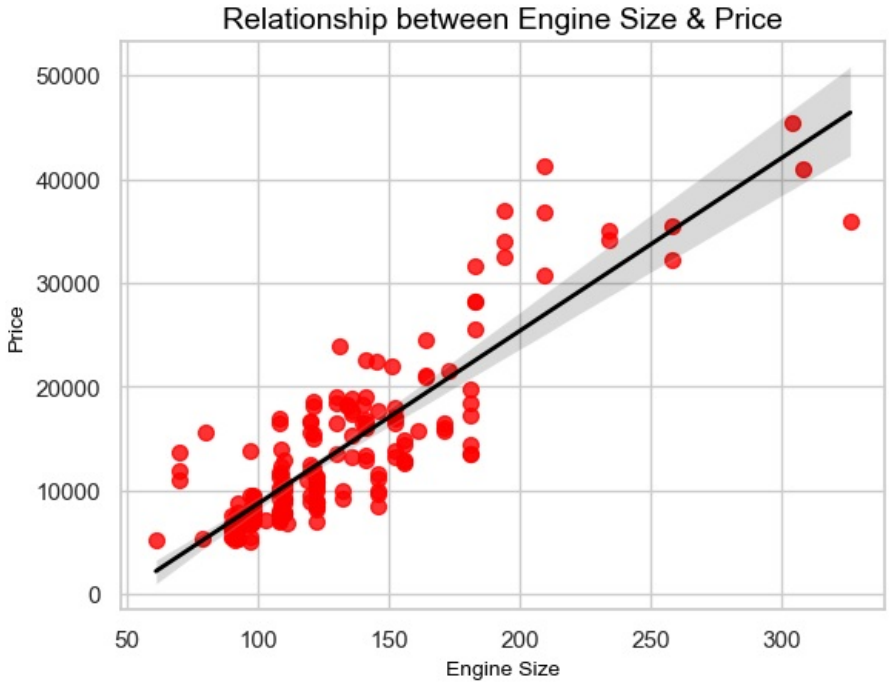
Insights: From this correlationshop between bore, stroke, compression-ratio and horsepower we see that

Bore size seems to have the most significant impact on horsepower. Compression ratio has a small negative effect on horsepower, while stroke length has an insignificant effect. Bore and stroke are largely independent of each other, and bore doesn't seem to influence

compression ratio either.

Find the scatterplot of "engine-size" and "price".

In [35]:
```
sns.regplot(x="engine-size", y="price", data=df, color="black", marker = "o", scatter_kws={"s": 50, "color": "r
            line_kws={"color" :"black", "linewidth" : 2})
plt.title("Relationship between Engine Size & Price", color = "black", fontsize = 14)
plt.xlabel("Engine Size", color = "Black", fontsize = 10)
plt.ylabel("Price" , color = "Black", fontsize =10)
plt.show()
```



Result:

As the engine-size goes up, the price goes up: this indicates apositive direct correlation between these two variables. Engine size seems like a pretty good predictorof price since the regression line is almost a perfect diagonal line.

Find the correlation between "engine-size" & "price"

In [39]:
```
df[["engine-size", "price"]].corr()
```

Out[39]:

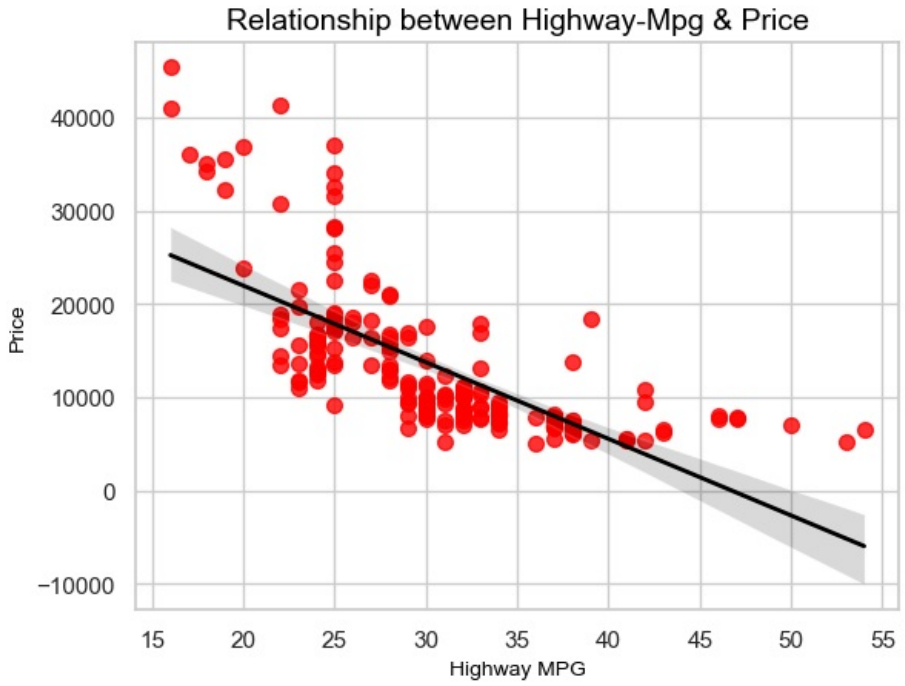
	engine-size	price
engine-size	1.000000	0.872335
price	0.872335	1.000000

Result:

The correlation between "engine-size", and "price" is .87 which shows a strong positive linear relationship.

Find the scatterplot of "highway-mpg" and "price".

In [49]:
```
sns.regplot(x = "highway-mpg", y = "price", data = df, color = "Black", marker = "o", scatter_kws={"s": 50, "co
            line_kws={"color" :"black", "linewidth" : 2})
plt.title("Relationship between Highway-Mpg & Price", color = "Black", fontsize = 14)
plt.xlabel("Highway MPG", color = "Black", fontsize = 10)
plt.ylabel("Price", color = "Black", fontsize = 10)
plt.show()
```



Result:

As highway-mpg goes up, the price goes down: this indicates an inverse/negative relationship between these two variables. Highway mpg could potentially be a predictor of price.

Find the Correlationship between "highway-mpg" & "price"

In [51]:
```
df[["highway-mpg", "price"]].corr()
```

Out[51]:

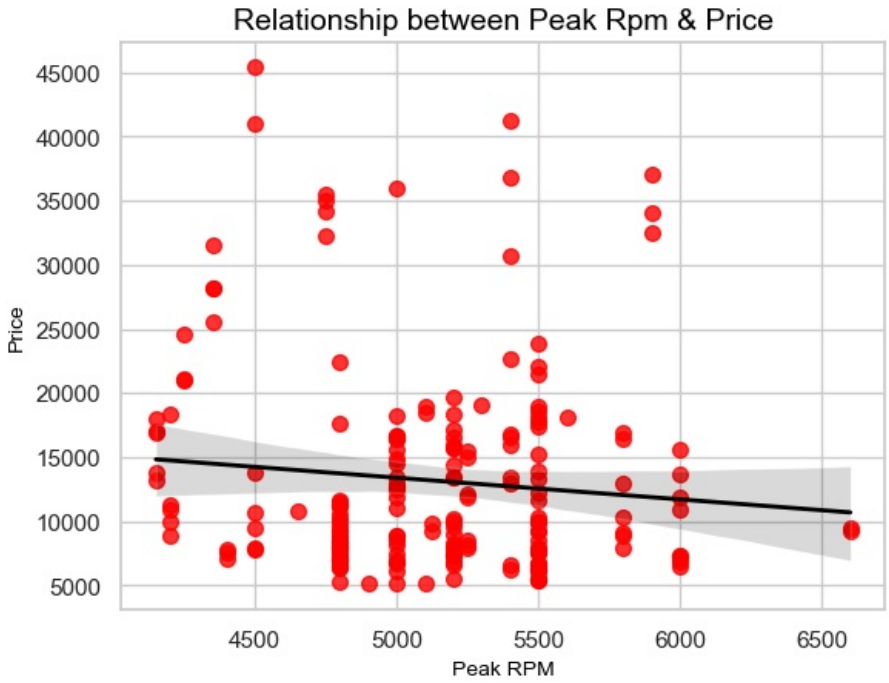
	highway-mpg	price
highway-mpg	1.000000	-0.704692
price	-0.704692	1.000000

Result:

The correlationshop between "highway-mpg" and "price is -0.70 which is strong negative linear relationship.

Find if the "peak-rpm" is a predictor variable of "price".

In [57]:
```
sns.regplot(x= "peak-rpm", y = "price", data=df, color = "Black", scatter_kws = {"s":50, "color":"Red"},
            line_kws={"color":"Black", "linewidth": 2 })
plt.title("Relationship between Peak Rpm & Price", color = "Black", fontsize = 14)
plt.xlabel("Peak RPM", color = "Black", fontsize = 10)
plt.ylabel("Price", color = "Black", fontsize = 10)
plt.show()
```



Result:

Peak rpm does not seem like a good predictor of the price at all since the regression line is close to horizontal. Also, the data points are very scattered and far from the fitted line, showing lots of variability. Therefore, it's not a reliable variable.

Find the correlation between "preak-rpm", "price"

In [61]:
```
df[["peak-rpm", "price"]].corr()
```

Out[61]:

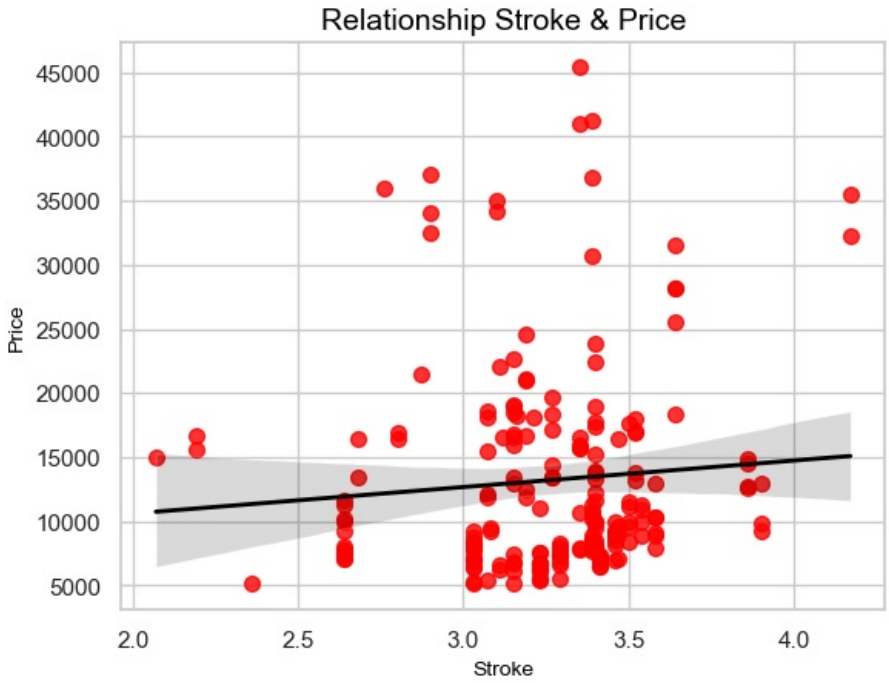
	peak-rpm	price
peak-rpm	1.000000	-0.101616
price	-0.101616	1.000000

Result:

The relationship is weak, suggesting that changes in the "peak-rpm" do not have a significant impact on "price".

Find the relationship between "stroke" & "price"

In [63]:
```
sns.regplot(x= "stroke", y = "price", data=df, color = "Black", scatter_kws = {"s":50, "color":"Red"},
            line_kws={"color":"Black", "linewidth": 2 })
plt.title("Relationship Stroke & Price", color = "Black", fontsize = 14)
plt.xlabel("Stroke", color = "Black", fontsize = 10)
plt.ylabel("Price", color = "Black", fontsize = 10)
plt.show()
```



Result:

Stroke does not seem like a good predictor of the price at all since the regression line is close to horizontal. Also, the data points are very scattered and far from the fitted line, showing lots of variability. Therefore, it's not a reliable variable.

Find the correlation between "stroke" & "price"

In [65]:
```
df[["stroke", "price"]].corr()
```

Out[65]:

	stroke	price
stroke	1.00000	0.08231
price	0.08231	1.00000

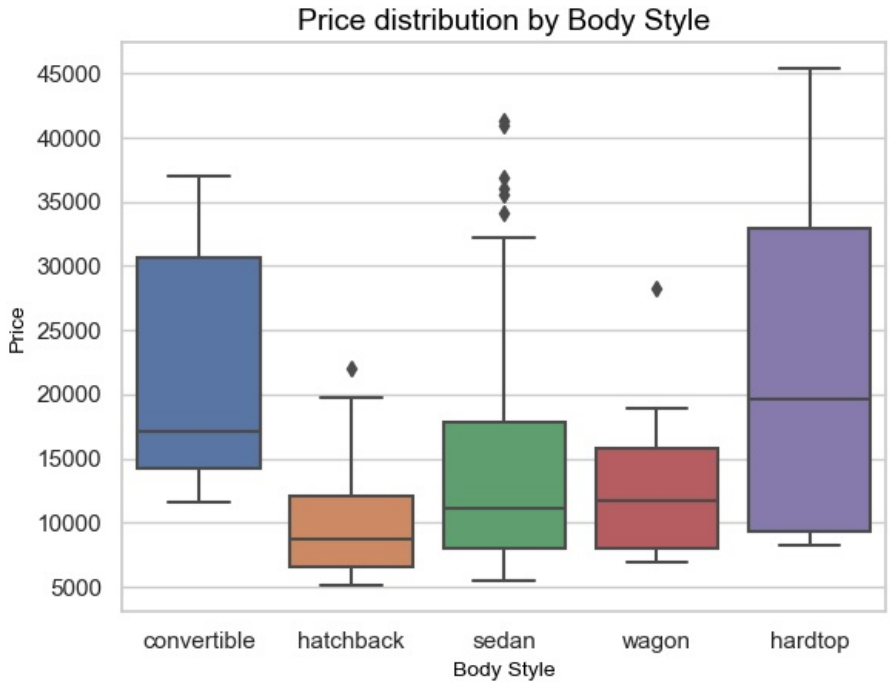
Result:

The relationship is weak, suggesting that changes in the "stroke" do not have a significant impact on "price".

Categorical Variables

Find the relationship between "body-style" and "price".

In [79]:
```
sns.boxplot(x = "body-style", y = "price", data = df)
plt.title( "Price distribution by Body Style", color = "black", fontsize = 14)
plt.xlabel("Body Style", color = "Black", fontsize = 10)
plt.ylabel("Price", color =  "Black", fontsize = 10)
plt.show()
```

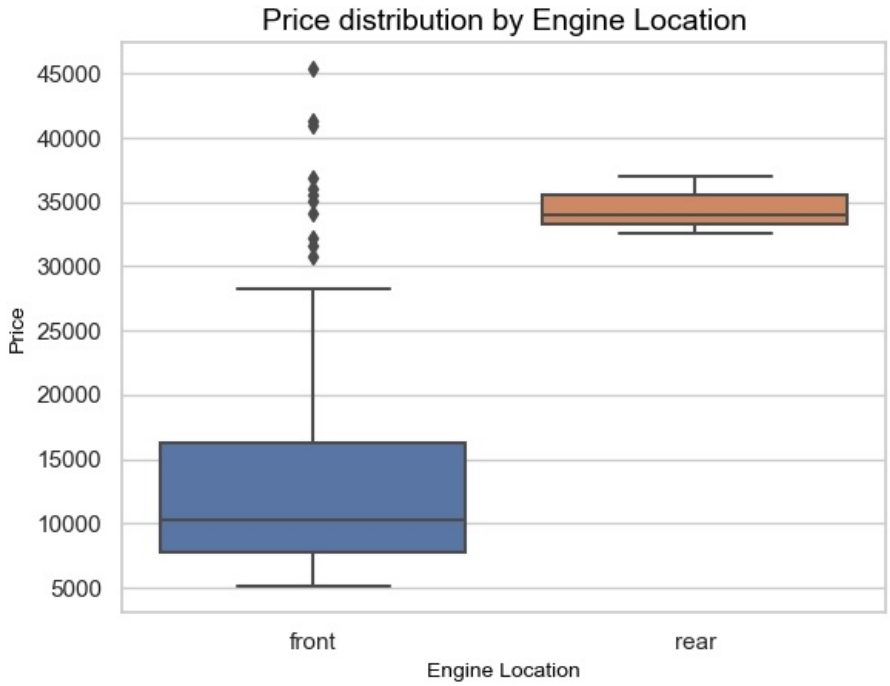


Result:

We see that the distributions of price between the different body-style categories have a significant overlap, so body-style would not be a good predictor of price.

Find the relationship between "engine-location", and "price".

In [81]:
```
sns.boxplot(x = "engine-location", y = "price", data=df)
plt.title( "Price distribution by Engine Location", color = "black", fontsize = 14)
plt.xlabel("Engine Location", color = "Black", fontsize = 10)
plt.ylabel("Price", color =  "Black", fontsize = 10)
plt.show()
```



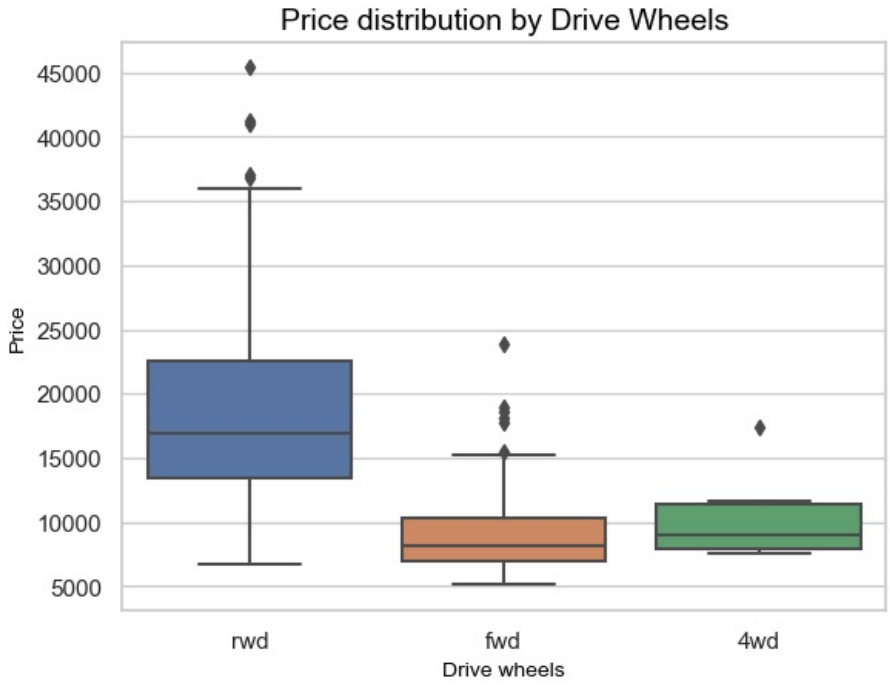
Result:

Here we see that the distribution of price between these two engine-location categories, front and rear, are distinct enough to take engine-location as a potential good predictor of price.

Find the relationship between "drive-wheel" and "price".

In [83]:
```
sns.boxplot(x = "drive-wheels", y = "price", data=df)
```

```
plt.title( "Price distribution by Drive Wheels", color = "black", fontsize = 14)
plt.xlabel("Drive wheels", color = "Black", fontsize = 10)
plt.ylabel("Price", color =  "Black", fontsize = 10)
plt.show()
```



Results:

Here we can see that price for rwd drive wheels differ from others two. so it can be a good predictor of price.

Descriptive Statistical Analysis

In [85]:
```
df.describe()
```

Out[85]:

	symboling	normalized-losses	wheel-base	length	width	height	curb-weight	engine-size	bore	stroke
count	201.000000	201.00000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	197.000000
mean	0.840796	122.00000	98.797015	0.837102	0.915126	53.766667	2555.666667	126.875622	3.330692	3.256904
std	1.254802	31.99625	6.066366	0.059213	0.029187	2.447822	517.296727	41.546834	0.268072	0.319256
min	-2.000000	65.00000	86.600000	0.678039	0.837500	47.800000	1488.000000	61.000000	2.540000	2.070000
25%	0.000000	101.00000	94.500000	0.801538	0.890278	52.000000	2169.000000	98.000000	3.150000	3.110000
50%	1.000000	122.00000	97.000000	0.832292	0.909722	54.100000	2414.000000	120.000000	3.310000	3.290000
75%	2.000000	137.00000	102.400000	0.881788	0.925000	55.500000	2926.000000	141.000000	3.580000	3.410000
max	3.000000	256.00000	120.900000	1.000000	1.000000	59.800000	4066.000000	326.000000	3.940000	4.170000

In [87]:
```
df.describe(include="object")
```

Out[87]:

	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

Value Counts

Create a Drive-wheel value count DataFrame

In [91]:
```
df["drive-wheels"].value_counts()
```

Out[91]:
```
drive-wheels
fwd    118
rwd     75
4wd      8
Name: count, dtype: int64
```

In [93]:
```
df["drive-wheels"].value_counts().to_frame()
```

Out[93]:

drive-wheels	count
fwd	118
rwd	75
4wd	8

In [101…
```
drive_wheel_count = df["drive-wheels"].value_counts().to_frame()
drive_wheel_count.reset_index(inplace = True)
drive_wheel_count.rename(columns={"drive-wheels": "Value Counts"}, inplace = True)
drive_wheel_count
```

Out[101…

	Value Counts	count
0	fwd	118
1	rwd	75
2	4wd	8

Create an Engine-Location value count DataFrame

In [107…
```
engine_location_value = df["engine-location"].value_counts().to_frame()
engine_location_value.reset_index(inplace = True)
engine_location_value.rename(columns = {"engine-location": "Value Counts"},inplace = True)
engine_location_value
```

Out[107…

	Value Counts	count
0	front	198
1	rear	3

Basics of Grouping

On average, which type of drive wheel is most valuable?

In [111…
```
grp_1 = df[["drive-wheels", "price"]]
df_grp = grp_1.groupby(["drive-wheels"], as_index = False).mean()
df_grp
```

Out[111…

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

Result:

From our data, it seems rear-wheel drive vehicles are, on average, the most expensive, while 4-wheel and front-wheel are approximately the same in price.

GroupBy Multiple Variable

In [113…
```
grp_2 = df[["drive-wheels", "body-style", "price"]]
df_grp_2 = grp_2.groupby(["drive-wheels", "body-style"], as_index = False).mean()
df_grp_2
```

Out[113…

	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

Create a Pivot table from grp_2

In [115…

```
grp_pivot = df_grp_2.pivot(index = "drive-wheels", columns = "body-style")
grp_pivot
```

Out[115…

	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

Fill Missing Data with 0 using fillna()

In [117…

```
grp_pivot = grp_pivot.fillna(0)
grp_pivot
```

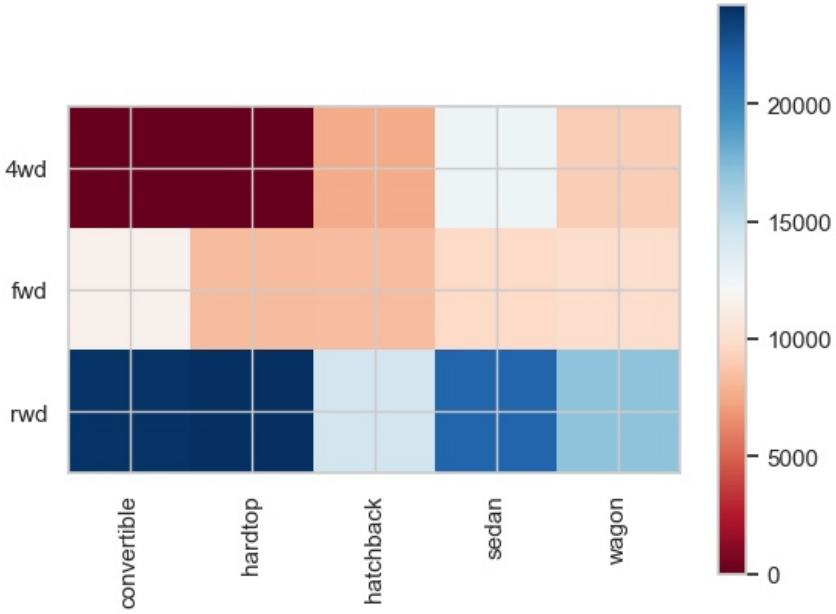
Out[117…

	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

Plot a Heatmap on grp_pivot

In [119…

```
fig, ax = plt.subplots()
im = ax.imshow(grp_pivot, cmap='RdBu')
ax.set_xticks(np.arange(grp_pivot.shape[1]))
ax.set_yticks(np.arange(grp_pivot.shape[0]))
ax.set_xticklabels(grp_pivot.columns.levels[1], rotation=90)
ax.set_yticklabels(grp_pivot.index)
fig.colorbar(im)
plt.show()
```



Correlation and Causation

Pearson Correlation

The Pearson Correlation measures the linear dependence between two variables X and Y.

The resulting coefficient is a value between -1 and 1 inclusive, where:

1: Perfect positive linear correlation. 0: No linear correlation, the two variables most likely do not affect each other. -1: Perfect negative linear correlation.

P-value

The P-value is the probability value that the correlation between these two variables is statistically significant. Normally, we choose a significance level of 0.05, which means that we are 95% confident that the correlation between the variables is significant.

By convention, when the

p-value is <0.001: we say there is strong evidence that the correlation is significant. the p-value is <0.05: there is moderate evidence that the correlation is significant. the p-value is <0.1: there is weak evidence that the correlation is significant. the p-value is >0.1: there is no evidence that the correlation is significant.

In [121…
```
from scipy import stats
```

Calculate the pearson coefficient and p-value on "wheel-base" and "price

In [123…
```
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 0.5846418222655081  with a P-value of P = 8.076488270732885e-20
```

Result:

Since the p-value is <0.001, the correlation between wheel-base and price is statistically significant, although the linear relationship isn't extremely strong (~0.585).

Calculate the Pearson Correlation Coefficient and P-value of 'horsepower' and 'price'.

In [131…
```
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 0.809574567003656 With a p-value of p =  6.369057428259557e-48
```

Result:

Since the p-value is < 0.001, the correlation between length and price is statistically significant, and the linear relationship is moderately strong (~0.691).

Calculate the Pearson Correlation Coefficient and P-value of 'width' and 'price'.

In [133…
```
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 0.7512653440522674  with a P-value of P = 9.20033551048144e-38
```

Result:

Since the p-value is < 0.001, the correlation between width and price is statistically significant, and the linear relationship is quite strong (~0.751).

Calculate the Pearson Correlation Coefficient and P-value of 'curb-weight' and 'price':

In [135…
```
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 0.8344145257702846  with a P-value of P =  2.1895772388937294e-53
```

Result:

Since the p-value is <0.001, the correlation between curb-weight and price is statistically significant, and the linear relationship is quite strong (~0.834).

Calculate the Pearson Correlation Coefficient and P-value of 'engine-size' and 'price':

In [137…
```
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
```

Result:

Since the p-value is <0.001, the correlation between engine-size and price is statistically significant, and the linear relationship is very strong (~0.872).

Calculate the Pearson Correlation Coefficient and P-value of 'bore' and 'price':

In [139…
```
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
```

Result:

Since the p-value is <0.001, the correlation between bore and price is statistically significant, but the linear relationship is only moderate (~0.521).

Calculate the Pearson Correlation Coefficient and P-value of 'city-mpg'and 'price':

In [141…
```
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 -0.6865710067844677  with a P-value of P =  2.3211320655676474e-29
```

Result:

Since the p-value is <0.001, the correlation between city-mpg and price is statistically significant, and the coefficient of about -0.687 shows that the relationship is negative and moderately strong.

Calculate the Pearson Correlation Coefficient and P-value of 'Highway-mpg'and 'price':

In [143…

```
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 -0.7046922650589529  with a P-value of P =  1.7495471144476463e-31
```

Result:

Since the p-value is < 0.001, the correlation between highway-mpg and price is statistically significant, and the coefficient of about -0.705 shows that the relationship is negative and moderately strong.