for data analysis

In [11]:
```
import pandas as pd
import numpy as np
```

for data visualization

In [12]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [13]:
```
df= pd.read_csv("/Users/Hp/AppData/Roaming/Microsoft/Windows/Start Menu/Programs/Anaconda3 (64-bit)/Data_Visualization_with_Python_s2-main/vw.csv")
display(df.shape)
print('number of rows=',df.shape[0])
print('number of columns=',df.shape[1])
display(df)
print(df)
ui=df.head()
display(ui)
```

```
(15157, 8)
number of rows= 15157
number of columns= 8
```

	model	year	price	transmission	mileage	fuelType	mpg	engineSize
0	T-Roc	2019	25000	Automatic	13904	Diesel	49.6	2.0
1	T-Roc	2019	26883	Automatic	4562	Diesel	49.6	2.0
2	T-Roc	2019	20000	Manual	7414	Diesel	50.4	2.0
3	T-Roc	2019	33492	Automatic	4825	Petrol	32.5	2.0
4	T-Roc	2019	22900	Semi-Auto	6500	Petrol	39.8	1.5
...	...	...	...	...	...	...	...	...
15152	Eos	2012	5990	Manual	74000	Diesel	58.9	2.0
15153	Fox	2008	1799	Manual	88102	Petrol	46.3	1.2
15154	Fox	2009	1590	Manual	70000	Petrol	42.0	1.4
15155	Fox	2006	1250	Manual	82704	Petrol	46.3	1.2
15156	Fox	2007	2295	Manual	74000	Petrol	46.3	1.2

15157 rows × 8 columns

```
        model  year  price transmission  mileage fuelType   mpg  engineSize
0       T-Roc  2019  25000    Automatic    13904   Diesel  49.6         2.0
1       T-Roc  2019  26883    Automatic     4562   Diesel  49.6         2.0
2       T-Roc  2019  20000       Manual     7414   Diesel  50.4         2.0
3       T-Roc  2019  33492    Automatic     4825   Petrol  32.5         2.0
4       T-Roc  2019  22900    Semi-Auto     6500   Petrol  39.8         1.5
...       ...   ...    ...          ...      ...      ...   ...         ...
15152     Eos  2012   5990       Manual    74000   Diesel  58.9         2.0
15153     Fox  2008   1799       Manual    88102   Petrol  46.3         1.2
15154     Fox  2009   1590       Manual    70000   Petrol  42.0         1.4
15155     Fox  2006   1250       Manual    82704   Petrol  46.3         1.2
15156     Fox  2007   2295       Manual    74000   Petrol  46.3         1.2

[15157 rows x 8 columns]
```

	model	year	price	transmission	mileage	fuelType	mpg	engineSize
0	T-Roc	2019	25000	Automatic	13904	Diesel	49.6	2.0
1	T-Roc	2019	26883	Automatic	4562	Diesel	49.6	2.0
2	T-Roc	2019	20000	Manual	7414	Diesel	50.4	2.0
3	T-Roc	2019	33492	Automatic	4825	Petrol	32.5	2.0
4	T-Roc	2019	22900	Semi-Auto	6500	Petrol	39.8	1.5

In [14]:
```
display(df.head())
```

	model	year	price	transmission	mileage	fuelType	mpg	engineSize
0	T-Roc	2019	25000	Automatic	13904	Diesel	49.6	2.0
1	T-Roc	2019	26883	Automatic	4562	Diesel	49.6	2.0
2	T-Roc	2019	20000	Manual	7414	Diesel	50.4	2.0
3	T-Roc	2019	33492	Automatic	4825	Petrol	32.5	2.0
4	T-Roc	2019	22900	Semi-Auto	6500	Petrol	39.8	1.5

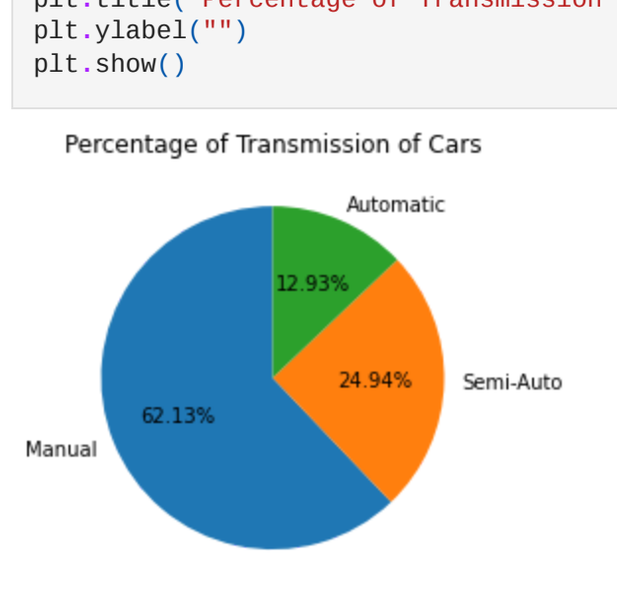
value counts()

In [15]:
```
display(df["transmission"].value_counts())
```

```
Manual       9417
Semi-Auto    3780
Automatic    1960
Name: transmission, dtype: int64
```

Pie-Chart

In [16]:
```
import matplotlib.pyplot as plt
import seaborn as sns
df["transmission"].value_counts().plot(kind="pie",autopct='%1.2f%%',startangle=90)
plt.title("Percentage of Transmission of Cars")
plt.ylabel("")
plt.show()
```



Creating DataFrame

In [17]:
```
dframe=pd.DataFrame(df["transmission"].value_counts())
display(dframe)
```

	transmission
Manual	9417
Semi-Auto	3780
Automatic	1960

In [18]:
```
dframe=dframe.reset_index()
dframe=dframe.rename(columns={"index":"transmission type",
                              "transmission":"number of cars"})
display(dframe)
```

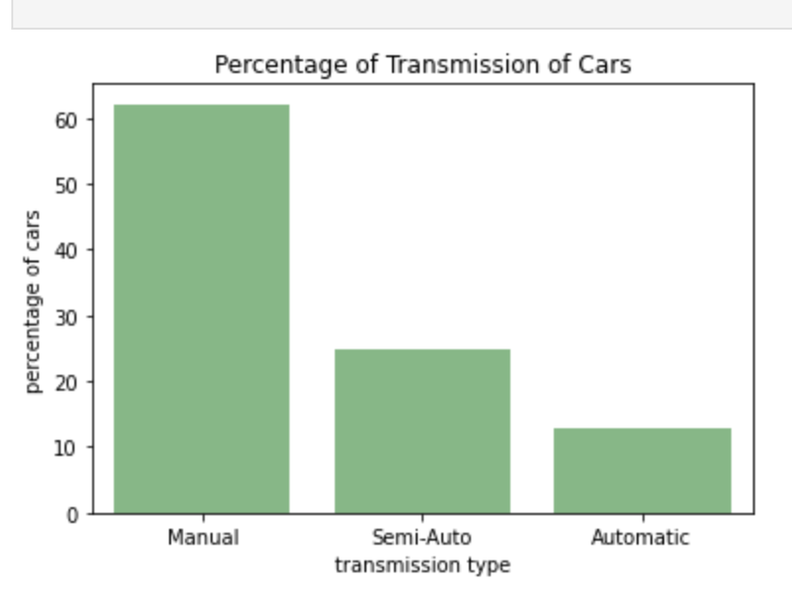
	transmission type	number of cars
0	Manual	9417
1	Semi-Auto	3780
2	Automatic	1960

In [19]:
```
dframe["percentage of cars"]=(dframe["number of cars"]/df.shape[0])*100
display(dframe.round(2))
```

	transmission type	number of cars	percentage of cars
0	Manual	9417	62.13
1	Semi-Auto	3780	24.94
2	Automatic	1960	12.93

Bar-plot

In [20]:
```
sns.barplot(x="transmission type",y="percentage of cars",data=dframe,alpha=0.50,color="green")
plt.title("Percentage of Transmission of Cars")
plt.show()
```



Comparison

The pie-chart gives a clear indication that manual type holds greater percentage alone than the the other two types combined. If needed, percentages can also be shown as annotations in the bar-chart but the pie-chart shows this comparison even without annotations. On the other hand, x and y-labels in the bar-graph presents a more detailed picture than the pie-chart as labels arev not used in pie-charts.

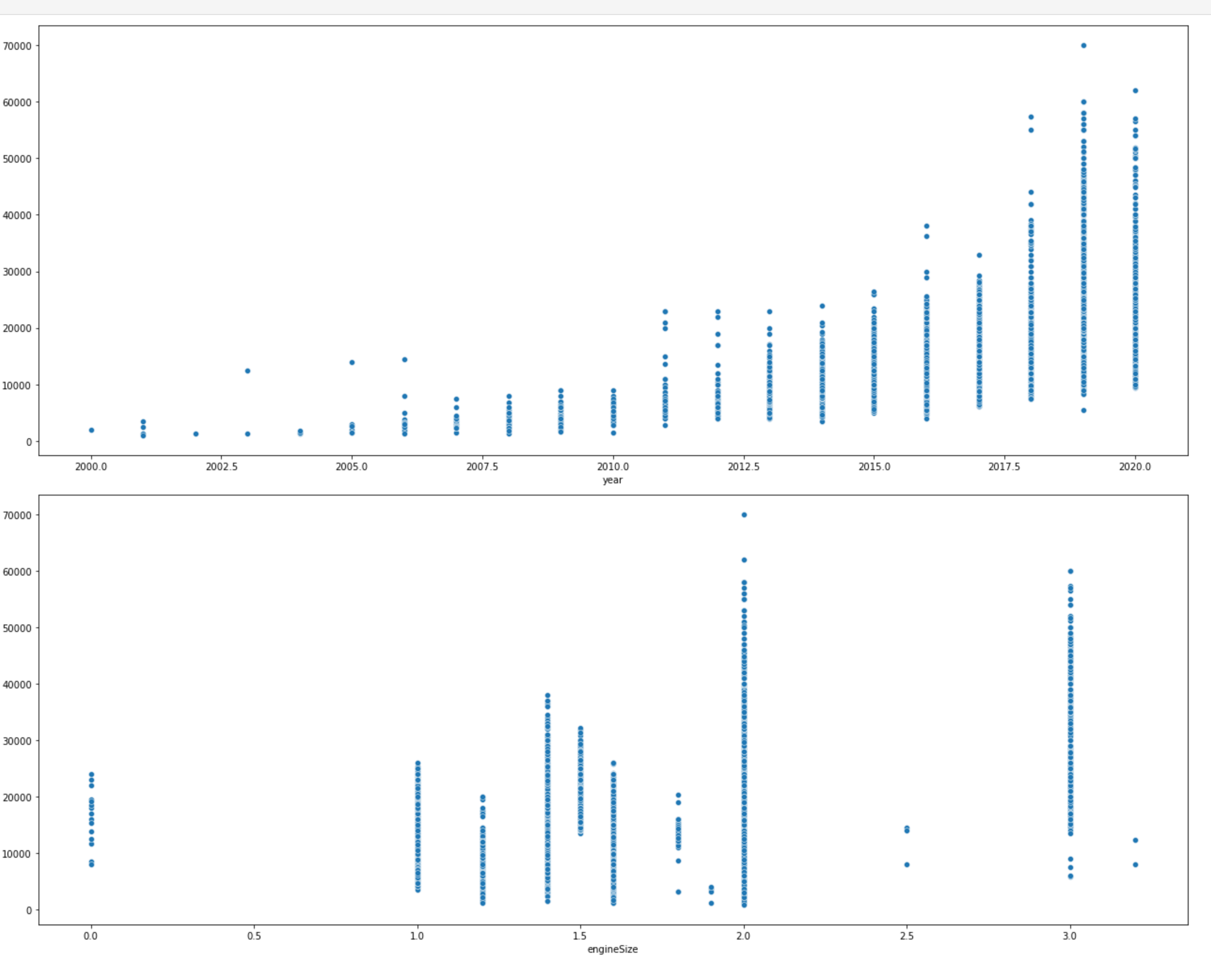
So, for a more detailed picture, bars can be helpful while pie-charts are applicable to make visualizations in an easier,faster way

In [21]:
```
def linear_eqn(x, m, c):
    x=np.arange(1,11,1)
    c=1
    y=m*x+c

    dframe = pd.DataFrame()

    dframe["x"] = x
    dframe["y"] = m*x + c


    display (dframe)
```

In [22]:
```
df= pd.read_csv("/Users/Hp/AppData/Roaming/Microsoft/Windows/Start Menu/Programs/Anaconda3 (64-bit)/Data_Visualization_with_Python_s2-main/vw.csv")
display(df.head(10))
```

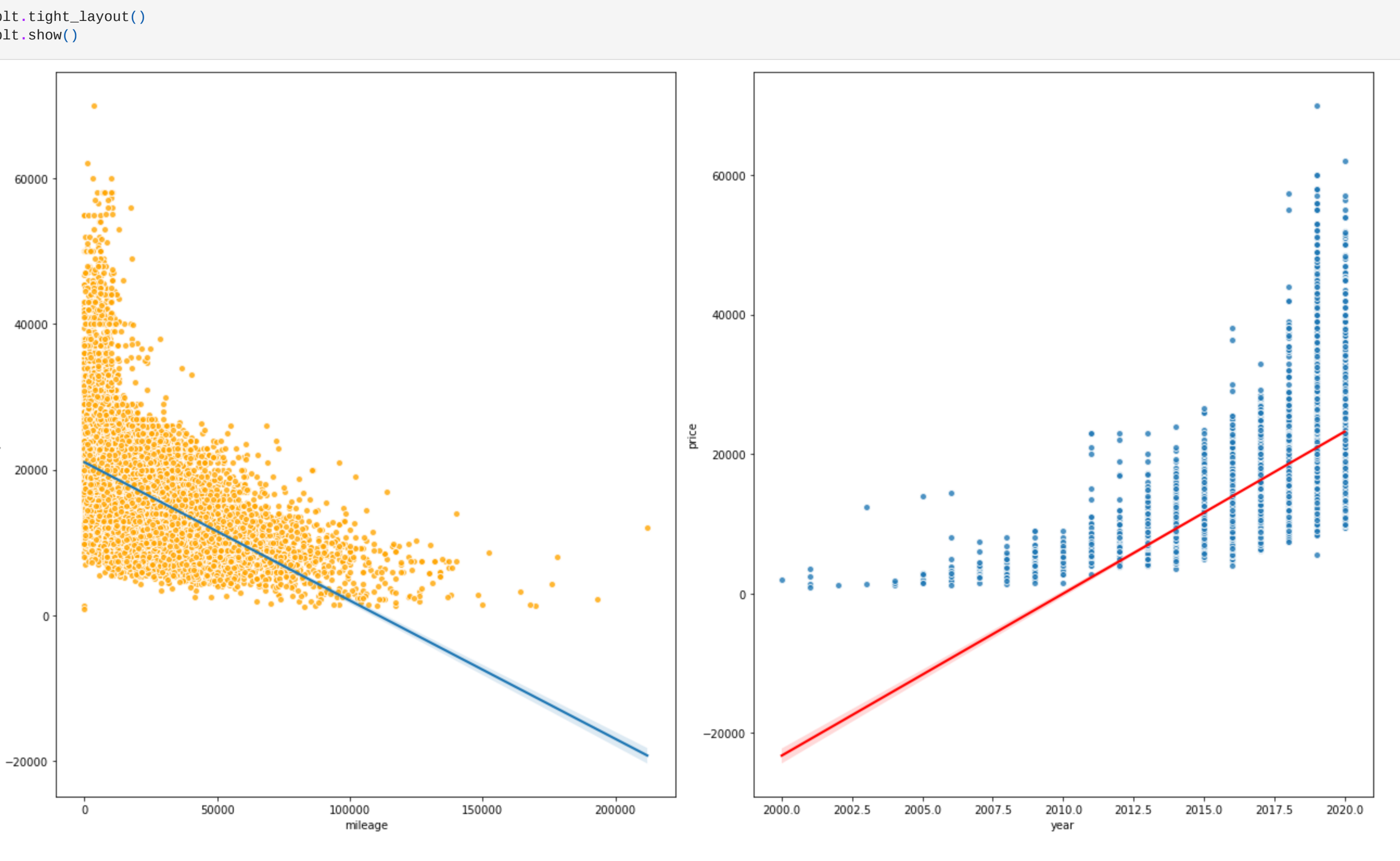
	model	year	price	transmission	mileage	fuelType	mpg	engineSize
0	T-Roc	2019	25000	Automatic	13904	Diesel	49.6	2.0
1	T-Roc	2019	26883	Automatic	4562	Diesel	49.6	2.0
2	T-Roc	2019	20000	Manual	7414	Diesel	50.4	2.0
3	T-Roc	2019	33492	Automatic	4825	Petrol	32.5	2.0
4	T-Roc	2019	22900	Semi-Auto	6500	Petrol	39.8	1.5
5	T-Roc	2020	31895	Manual	10	Petrol	42.2	1.5
6	T-Roc	2020	27895	Manual	10	Petrol	42.2	1.5
7	T-Roc	2020	39495	Semi-Auto	10	Petrol	32.5	2.0
8	T-Roc	2019	21995	Manual	10	Petrol	44.1	1.0
9	T-Roc	2019	23285	Manual	10	Petrol	42.2	1.5

In [23]:
```
plt.figure(figsize=(18,14))

plt.subplot(2, 1, 1)
sns.scatterplot(x="year", y="price", data=df)



plt.subplot(2, 1, 2)
sns.scatterplot(x="engineSize", y="price", data=df)

plt.tight_layout()
plt.show()
```



In the price Vs. year scatter plot, it is following a certain trend as the price rises wuth the passing of years. But the plot doesn't show any distinct relationship between engineSize and price. Price sometimes rises, sometimes stays constant, even sometimes falls as the increase in engine size.

In [24]:
```
plt.figure(figsize=(18,10))

plt.subplot(1,2, 1)
sns.regplot(x="mileage", y="price", data=df,scatter_kws={"color":"orange", "edgecolor":"white"})


plt.subplot(1, 2, 2)
sns.regplot(x="year",y="price", data=df, line_kws={"color":"red"},scatter_kws={"edgecolor":"white"})



plt.tight_layout()
plt.show()
```



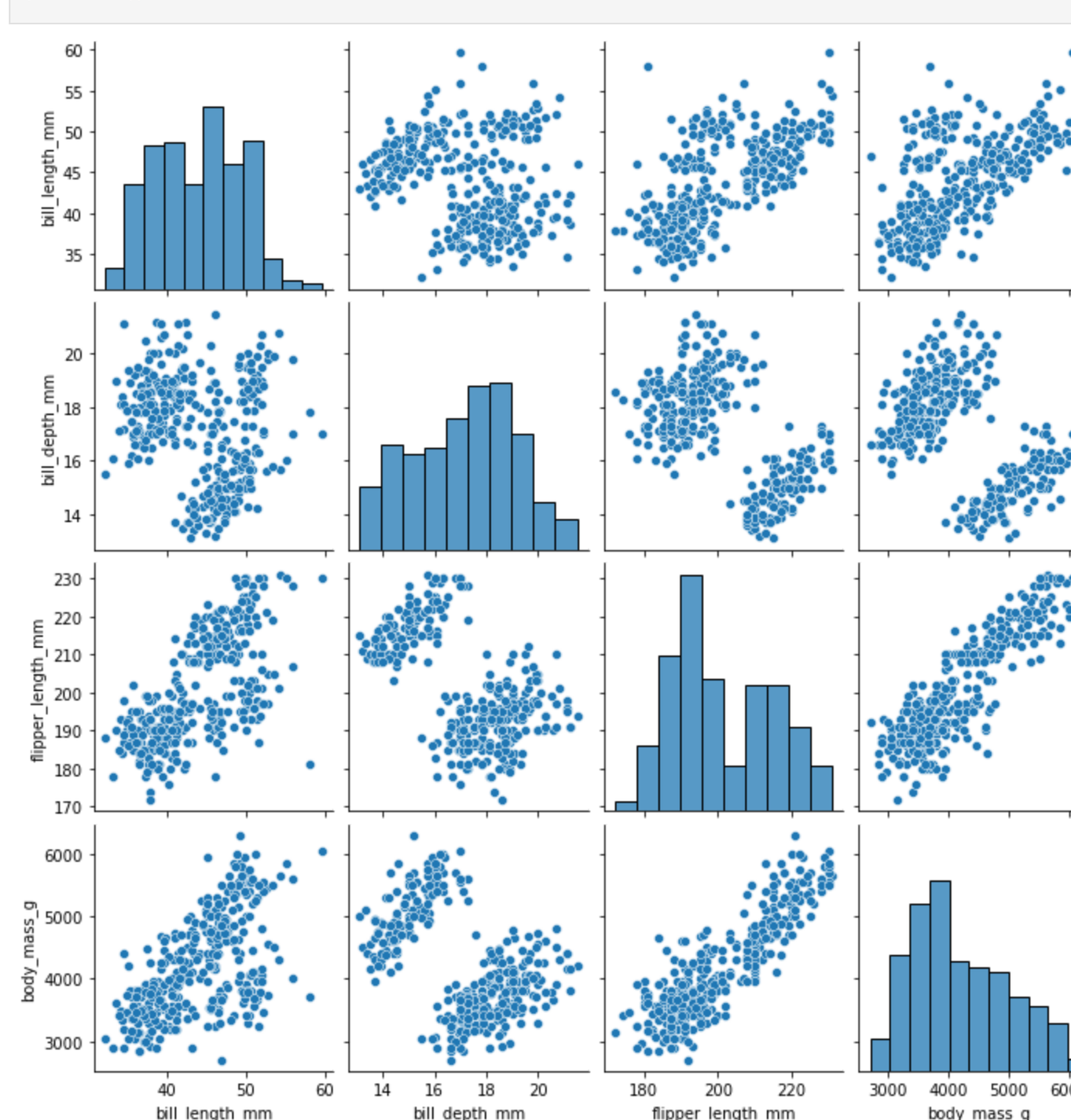
The regression plot also helps us to undertand the underlying relation between dependent and independent variables, here, between mileage and price and then year and price. As the graph indicates, price falls with the increase in mileage and rises with the passing of year. The trrwegression line helps the audience to understand the trend.

In [25]:
```
penguins = sns.load_dataset("penguins")

display(penguins.head())
print(penguins.shape)
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female

```
(344, 7)
```

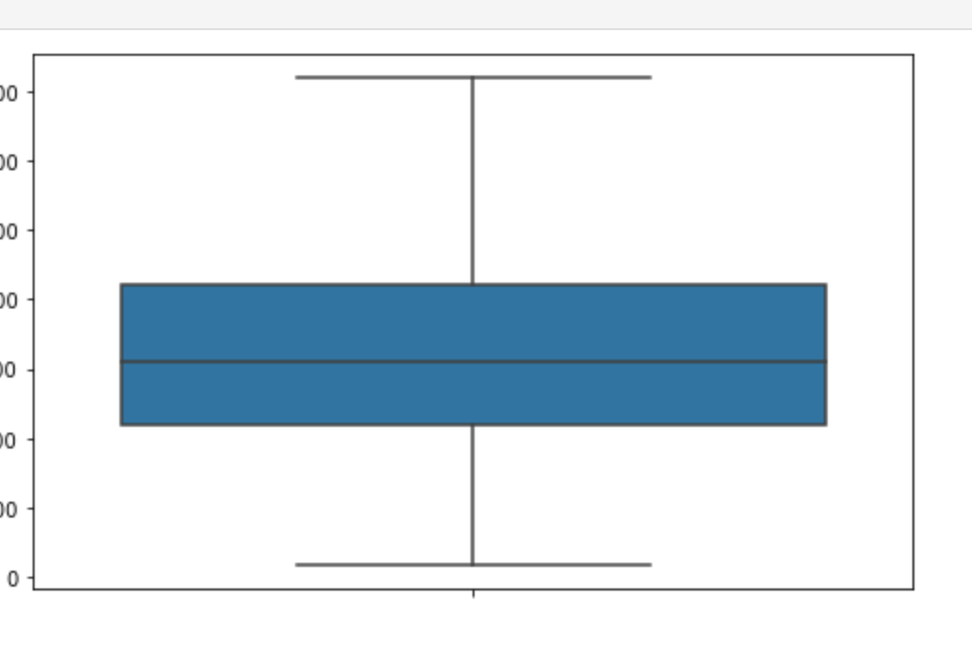
In [26]:
```
sns.pairplot(penguins)
plt.show()
```



In [27]:
```
sns.pairplot(penguins, hue="species")
plt.show()
```

Pair-plot helps to visualize multiple inputs in a single plot. This also helps to compare those inputs among themselves. For example, here, it is shown that in some cases, Chinstrap and Adelie show same characteristics as blue and orange dots are overlapping while the green dots are showing distant characters. In mostv cases, relation between x and y axis is positive. Adding a categorical column as hue was important as different color points indicate different inputs.

In [28]:
```
plt.figure(figsize=(8,5))
sns.boxplot(y = "price", data=df, showfliers=False)
plt.show()
```



Box-plot gives a summary of the visualization and can handle large amount of data. Also, it indicates the min and max values of the selected data. But, it comes with some disadvantages. It cannot return the exact value of given the data.

In [29]:
```
df.columns
```

Out[29]:
```
Index(['model', 'year', 'price', 'transmission', 'mileage', 'fuelType', 'mpg',
       'engineSize'],
      dtype='object')
```

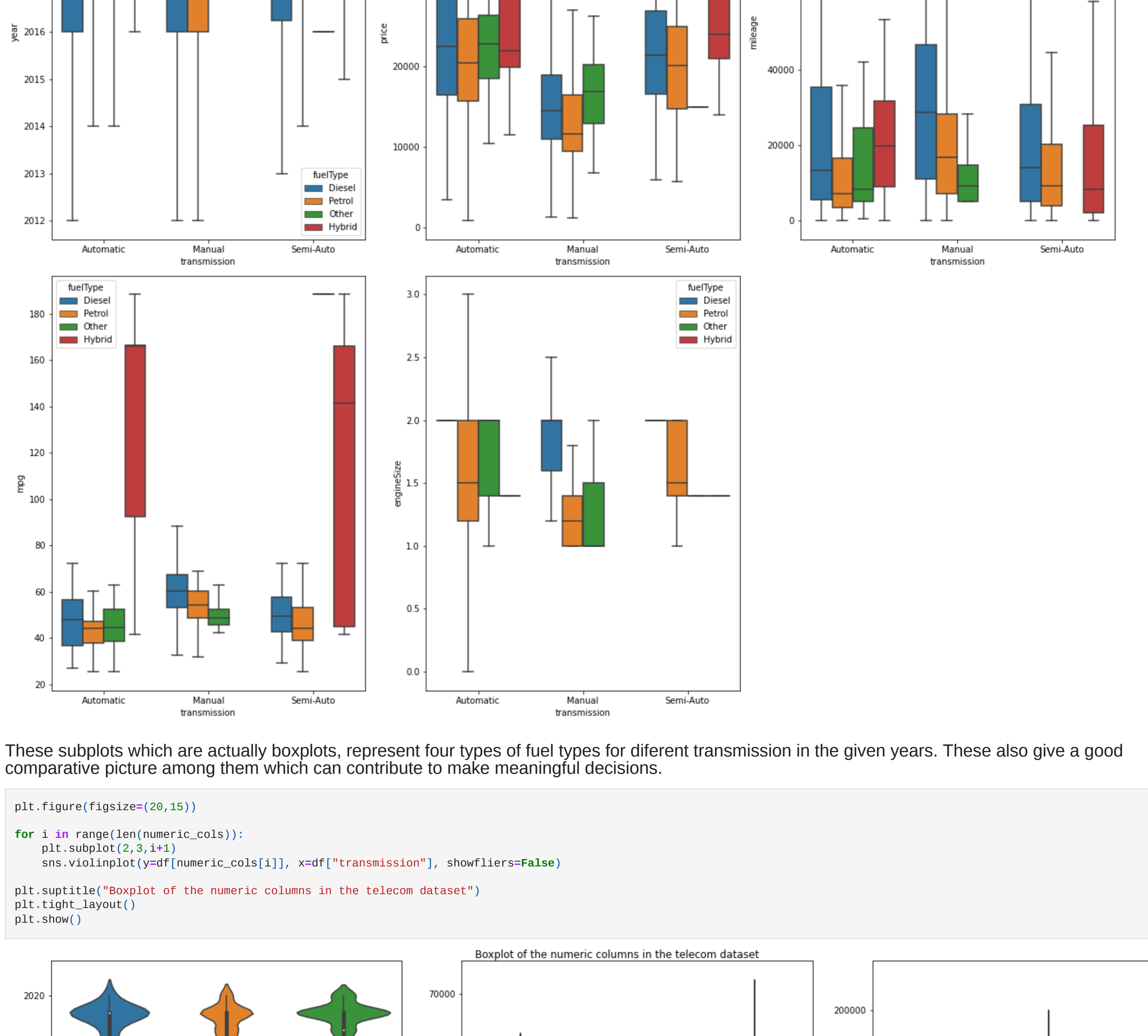
In [30]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15157 entries, 0 to 15156
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   model         15157 non-null  object 
 1   year          15157 non-null  int64  
 2   price         15157 non-null  int64  
 3   transmission  15157 non-null  object 
 4   mileage       15157 non-null  int64  
 5   fuelType      15157 non-null  object 
 6   mpg           15157 non-null  float64
 7   engineSize    15157 non-null  float64
dtypes: float64(2), int64(3), object(3)
memory usage: 947.4+ KB
```

In [31]:
```
numeric_cols = []

for colm in df.columns:
    if df[colm].dtypes != "O":
        numeric_cols.append(colm)


print(numeric_cols)
```

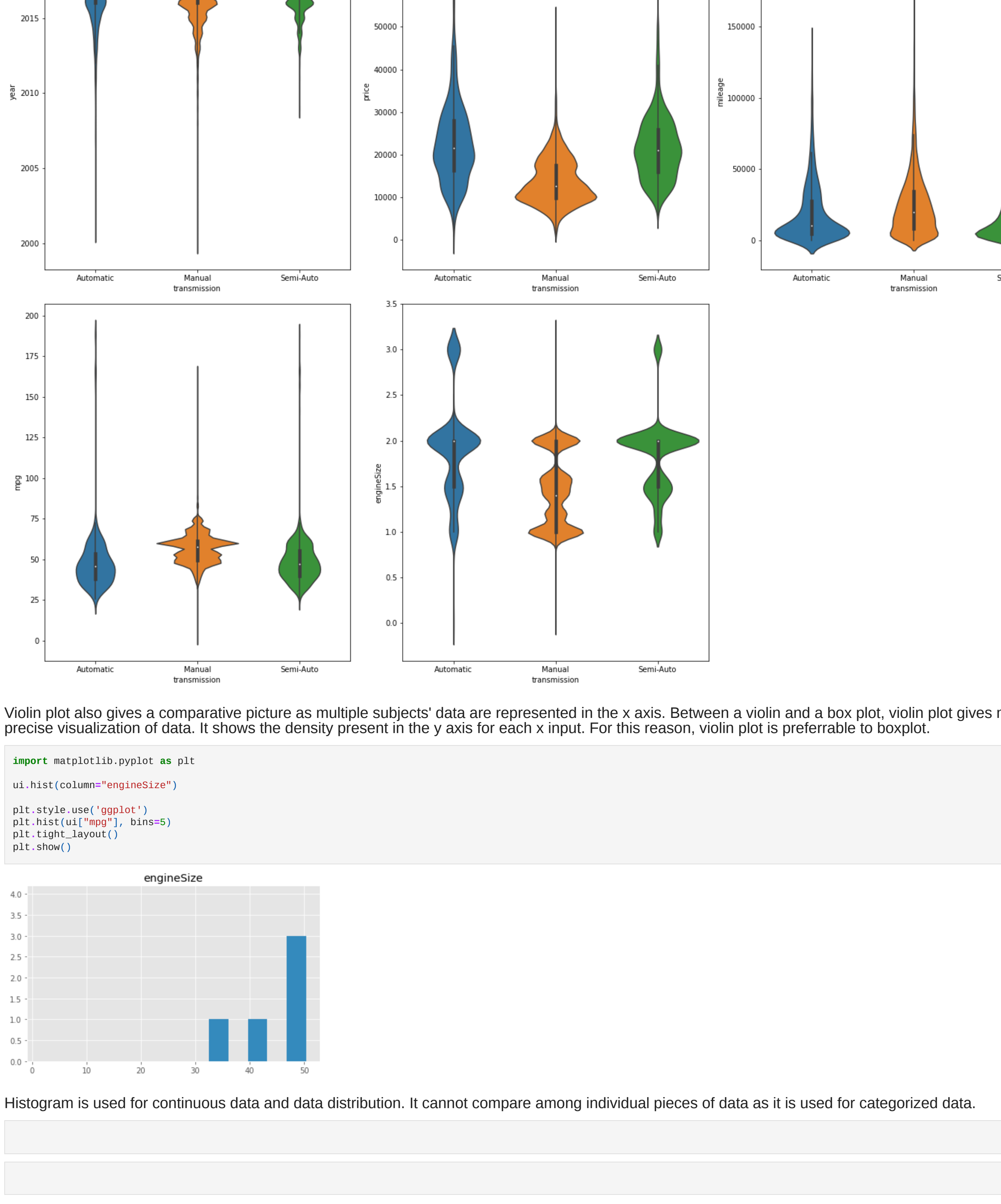
```
['year', 'price', 'mileage', 'mpg', 'engineSize']
```

In [32]:
```
plt.figure(figsize=(18,15))

for i in range(len(numeric_cols)):
    plt.subplot(2,3,i+1)
    sns.boxplot(y=df[numeric_cols[i]], x=df["transmission"], hue=df["fuelType"], showfliers=False)

plt.suptitle("Boxplot of the transmission & fuelType columns in the vw dataset")
plt.tight_layout()
plt.show()
```



These subplots which are actually boxplots, represent four types of fuel types for diferent transmission in the given years. These also give a good comparative picture among them which can contribute to make meaningful decisions.

In [33]:
```
plt.figure(figsize=(20,15))

for i in range(len(numeric_cols)):
    plt.subplot(2,3,i+1)
    sns.violinplot(y=df[numeric_cols[i]], x=df["transmission"], showfliers=False)

plt.suptitle("Boxplot of the numeric columns in the telecom dataset")
plt.tight_layout()
plt.show()
```



Violin plot also gives a comparative picture as multiple subjects' data are represented in the x axis. Between a violin and a box plot, violin plot gives more precise visualization of data. It shows the density present in the y axis for each x input. For this reason, violin plot is preferrable to boxplot.

In [38]:
```
import matplotlib.pyplot as plt

ui.hist(column="engineSize")

plt.style.use('ggplot')
plt.hist(ui["mpg"], bins=5)
plt.tight_layout()
plt.show()
```


Histogram is used for continuous data and data distribution. It cannot compare among individual pieces of data as it is used for categorized data.

In [ ]:

In [ ]: