Practical No: 7

Topics Covered: Pandas, Matplotlib

Name: Saloni Vishwakarma

Batch-Roll no: C1-13

A] Write the Python code apply Data Cleaning Techniques for the following.

1) Load the Toyota vehicle's data from a given csv file into a data frame and print it. (Toyota.csv)

In [28]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [29]:
```
datafile="Toyota.csv"
```

In [30]:
```
dataset=pd.read_csv(datafile)
dataset
```

Out[30]:

	Unnamed: 0	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	0	13500	23.0	46986	Diesel	90	1.0	0	2000	three	1165
1	1	13750	23.0	72937	Diesel	90	1.0	0	2000	3	1165
2	2	13950	24.0	41711	Diesel	90	NaN	0	2000	3	1165
3	3	14950	26.0	48000	Diesel	90	0.0	0	2000	3	1165
4	4	13750	30.0	38500	Diesel	90	0.0	0	2000	3	1170
...	...	...	...	...	...	...	...	...	...	...	...
1431	1431	7500	NaN	20544	Petrol	86	1.0	0	1300	3	1025
1432	1432	10845	72.0	??	Petrol	86	0.0	0	1300	3	1015
1433	1433	8500	NaN	17016	Petrol	86	0.0	0	1300	3	1015
1434	1434	7250	70.0	??	NaN	86	1.0	0	1300	3	1015
1435	1435	6950	76.0	1	Petrol	110	0.0	0	1600	5	1114

1436 rows × 11 columns

2) Replace all abnormal symbols such as '?' '***' etc. by Null values

In [55]:
```
values=['?','??','***']
df=dataset.replace(values,np.NaN)
df
```

Out[55]:

	Unnamed: 0	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	0	13500	23.0	46986	Diesel	90	1.0	0	2000	three	1165
1	1	13750	23.0	72937	Diesel	90	1.0	0	2000	3	1165
2	2	13950	24.0	41711	Diesel	90	NaN	0	2000	3	1165
3	3	14950	26.0	48000	Diesel	90	0.0	0	2000	3	1165
4	4	13750	30.0	38500	Diesel	90	0.0	0	2000	3	1170
...	...	...	...	...	...	...	...	...	...	...	...
1431	1431	7500	60.0	20544	Petrol	86	1.0	0	1300	3	1025
1432	1432	10845	72.0	NaN	Petrol	86	0.0	0	1300	3	1015
1433	1433	8500	60.0	17016	Petrol	86	0.0	0	1300	3	1015
1434	1434	7250	70.0	NaN	NaN	86	1.0	0	1300	3	1015
1435	1435	6950	76.0	1	Petrol	110	0.0	0	1600	5	1114

1436 rows × 11 columns

3) Count the total number of null values present in each attribute (column)

In [32]:
```
df.isnull().sum()
```

Out[32]:
```
Unnamed: 0      0
Price           0
Age           100
KM             15
FuelType      100
HP              0
MetColor      150
Automatic       0
CC              0
Doors           0
Weight          0
dtype: int64
```

4) Replace the NaN in Age column by median value.

In [33]:
```
dataset=pd.read_csv("Toyota.csv")
x=dataset["Age"].median()
print('Median=',x)
dataset["Age"].fillna(x,inplace=True)
dataset
```

```
Median= 60.0
```

Out[33]:

	Unnamed: 0	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	0	13500	23.0	46986	Diesel	90	1.0	0	2000	three	1165
1	1	13750	23.0	72937	Diesel	90	1.0	0	2000	3	1165
2	2	13950	24.0	41711	Diesel	90	NaN	0	2000	3	1165
3	3	14950	26.0	48000	Diesel	90	0.0	0	2000	3	1165
4	4	13750	30.0	38500	Diesel	90	0.0	0	2000	3	1170
...	...	...	...	...	...	...	...	...	...	...	...
1431	1431	7500	60.0	20544	Petrol	86	1.0	0	1300	3	1025
1432	1432	10845	72.0	??	Petrol	86	0.0	0	1300	3	1015
1433	1433	8500	60.0	17016	Petrol	86	0.0	0	1300	3	1015
1434	1434	7250	70.0	??	NaN	86	1.0	0	1300	3	1015
1435	1435	6950	76.0	1	Petrol	110	0.0	0	1600	5	1114

1436 rows × 11 columns

5) Replace the string value in column Door with number.

In [34]:
```
df.loc[0,'Doors']=3
df
```

Out[34]:

	Unnamed: 0	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	0	13500	23.0	46986	Diesel	90	1.0	0	2000	3	1165
1	1	13750	23.0	72937	Diesel	90	1.0	0	2000	3	1165
2	2	13950	24.0	41711	Diesel	90	NaN	0	2000	3	1165
3	3	14950	26.0	48000	Diesel	90	0.0	0	2000	3	1165
4	4	13750	30.0	38500	Diesel	90	0.0	0	2000	3	1170
...	...	...	...	...	...	...	...	...	...	...	...
1431	1431	7500	NaN	20544	Petrol	86	1.0	0	1300	3	1025
1432	1432	10845	72.0	NaN	Petrol	86	0.0	0	1300	3	1015
1433	1433	8500	NaN	17016	Petrol	86	0.0	0	1300	3	1015
1434	1434	7250	70.0	NaN	NaN	86	1.0	0	1300	3	1015
1435	1435	6950	76.0	1	Petrol	110	0.0	0	1600	5	1114

1436 rows × 11 columns

6) Returns a Series containing counts of unique values for column ‘FuelType’ (Petrol, Diesel, CNG)

In [35]:
```
print(df['FuelType'].value_counts())
```

```
Petrol    1177
Diesel     144
CNG         15
Name: FuelType, dtype: int64
```

7) Get the mode value of FuelType and fill NA/NaN values using the specified value of mode.

In [36]:
```
m=df['FuelType'].mode()[0]
print(m)
df['FuelType'].fillna(m,inplace=True)
df
```

```
Petrol
```

Out[36]:

	Unnamed: 0	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	0	13500	23.0	46986	Diesel	90	1.0	0	2000	3	1165
1	1	13750	23.0	72937	Diesel	90	1.0	0	2000	3	1165
2	2	13950	24.0	41711	Diesel	90	NaN	0	2000	3	1165
3	3	14950	26.0	48000	Diesel	90	0.0	0	2000	3	1165
4	4	13750	30.0	38500	Diesel	90	0.0	0	2000	3	1170
...	...	...	...	...	...	...	...	...	...	...	...
1431	1431	7500	NaN	20544	Petrol	86	1.0	0	1300	3	1025
1432	1432	10845	72.0	NaN	Petrol	86	0.0	0	1300	3	1015
1433	1433	8500	NaN	17016	Petrol	86	0.0	0	1300	3	1015
1434	1434	7250	70.0	NaN	Petrol	86	1.0	0	1300	3	1015
1435	1435	6950	76.0	1	Petrol	110	0.0	0	1600	5	1114

1436 rows × 11 columns

8) Get the mode value of Metcolor and fill NA/NaN values using the specified value of mode.

In [37]:
```
m=df['MetColor'].mode()[0]
print(m)
df['MetColor'].fillna(m,inplace=True)
df
```

```
1.0
```

Out[37]:

	Unnamed: 0	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	0	13500	23.0	46986	Diesel	90	1.0	0	2000	3	1165
1	1	13750	23.0	72937	Diesel	90	1.0	0	2000	3	1165
2	2	13950	24.0	41711	Diesel	90	1.0	0	2000	3	1165
3	3	14950	26.0	48000	Diesel	90	0.0	0	2000	3	1165
4	4	13750	30.0	38500	Diesel	90	0.0	0	2000	3	1170
...	...	...	...	...	...	...	...	...	...	...	...
1431	1431	7500	NaN	20544	Petrol	86	1.0	0	1300	3	1025
1432	1432	10845	72.0	NaN	Petrol	86	0.0	0	1300	3	1015
1433	1433	8500	NaN	17016	Petrol	86	0.0	0	1300	3	1015
1434	1434	7250	70.0	NaN	Petrol	86	1.0	0	1300	3	1015
1435	1435	6950	76.0	1	Petrol	110	0.0	0	1600	5	1114

1436 rows × 11 columns

9) Replace the value of FuelType as for Petrol replaces with 0, diesel with 1 and CNG with 2.

In [15]:
```
df['FuelType'].replace({'petrol':0,'diesel':1,'CNG':2},inplace=True)
df
```

Out[15]:

	Unnamed: 0	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	0	13500	23.0	46986	Diesel	90	1.0	0	2000	0	1165
1	1	13750	23.0	72937	Diesel	90	1.0	0	2000	3	1165
2	2	13950	24.0	41711	Diesel	90	1.0	0	2000	3	1165
3	3	14950	26.0	48000	Diesel	90	0.0	0	2000	3	1165
4	4	13750	30.0	38500	Diesel	90	0.0	0	2000	3	1170
...	...	...	...	...	...	...	...	...	...	...	...
1431	1431	7500	NaN	20544	Petrol	86	1.0	0	1300	3	1025
1432	1432	10845	72.0	??	Petrol	86	0.0	0	1300	3	1015
1433	1433	8500	NaN	17016	Petrol	86	0.0	0	1300	3	1015
1434	1434	7250	70.0	??	Petrol	86	1.0	0	1300	3	1015
1435	1435	6950	76.0	1	Petrol	110	0.0	0	1600	5	1114

1436 rows × 11 columns

B] Analysis of Automobile dataset (Automobile_data.csv)

1. Load the Automobile data from a given csv file into a data frame and print the shape of the data, type of the data and first and last 4 rows of dataset. (Automobile_data.csv)

In [16]:
```
datafile="Automobile_data.csv"
dataset=pd.read_csv(datafile)
dataset
```

Out[16]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	whe ba
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	8
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	8
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	9
3	2	164	audi	gas	std	four	sedan	fwd	front	9
4	2	164	audi	gas	std	four	sedan	4wd	front	9
...	...	...	...	...	...	...	...	...	...	
200	-1	95	volvo	gas	std	four	sedan	rwd	front	10
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	10
202	-1	95	volvo	gas	std	four	sedan	rwd	front	10
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	10
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	10

205 rows × 26 columns

In [17]:
```
#to print first 4 rows
dataset.head(4)
```

Out[17]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8

4 rows × 26 columns

In [18]:
```
#to print last 4 rows
dataset.tail(4)
```

Out[18]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	..
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	..
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	..
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1	..
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	..

4 rows × 26 columns

In [19]:
```
#to check the type of data
dataset.dtypes
```

Out[19]:
```
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 [20]:
```
#to check the shape of data
dataset.shape
```

Out[20]:
```
(205, 26)
```

2. Replace the abnormal symbols i.e. ??, ## to null value i.e., NaN

In [23]:
```
values=['?','??','##','***']
df=dataset.replace(values,np.NaN)
df
```

Out[23]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	whe ba
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	8
1	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	8
2	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front	9
3	2	164	audi	gas	std	four	sedan	fwd	front	9
4	2	164	audi	gas	std	four	sedan	4wd	front	9
...	...	...	...	...	...	...	...	...	...	
200	-1	95	volvo	gas	std	four	sedan	rwd	front	10
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	10
202	-1	95	volvo	gas	std	four	sedan	rwd	front	10
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	10
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	10

205 rows × 26 columns

3. Describe the dataset, and find number of observations, missing values and nan values.

In [18]:
```
#to describe the dataset
print(dataset.describe)
```

```
<bound method NDFrame.describe of      symboling normalized-losses         ma
ke fuel-type aspiration  \
0            3                 ?  alfa-romero       gas        std
1            3                 ?  alfa-romero       gas        std
2            1                 ?  alfa-romero       gas        std
3            2               164         audi       gas        std
4            2               164         audi       gas        std
..         ...               ...          ...       ...        ...
200         -1                95        volvo       gas        std
201         -1                95        volvo       gas      turbo
202         -1                95        volvo       gas        std
203         -1                95        volvo    diesel      turbo
204         -1                95        volvo       gas      turbo

    num-of-doors   body-style drive-wheels engine-location  wheel-base  ...
\
0            two  convertible          rwd           front        88.6  ...
1            two  convertible          rwd           front        88.6  ...
2            two    hatchback          rwd           front        94.5  ...
3           four        sedan          fwd           front        99.8  ...
4           four        sedan          4wd           front        99.4  ...
..           ...          ...          ...             ...         ...  ...
200         four        sedan          rwd           front       109.1  ...
201         four        sedan          rwd           front       109.1  ...
202         four        sedan          rwd           front       109.1  ...
203         four        sedan          rwd           front       109.1  ...
204         four        sedan          rwd           front       109.1  ...

     engine-size  fuel-system  bore  stroke compression-ratio horsepower  \
0            130         mpfi  3.47    2.68               9.0        111
1            130         mpfi  3.47    2.68               9.0        111
2            152         mpfi  2.68    3.47               9.0        154
3            109         mpfi  3.19     3.4              10.0        102
4            136         mpfi  3.19     3.4               8.0        115
..           ...          ...   ...     ...               ...        ...
200          141         mpfi  3.78    3.15               9.5        114
201          141         mpfi  3.78    3.15               8.7        160
202          173         mpfi  3.58    2.87               8.8        134
203          145          idi  3.01     3.4              23.0        106
204          141         mpfi  3.78    3.15               9.5        114

     peak-rpm city-mpg highway-mpg  price
0        5000       21          27  13495
1        5000       21          27  16500
2        5000       19          26  16500
3        5500       24          30  13950
4        5500       18          22  17450
..        ...      ...         ...    ...
200      5400       23          28  16845
201      5300       19          25  19045
202      5500       18          23  21485
203      4800       26          27  22470
204      5400       19          25  22625
```

```
[205 rows x 26 columns]>
```

In [19]:
```
#to count the number of observations
len(dataset)
```

Out[19]:
```
205
```

In [38]:
```
dataset.isnull()
```

Out[38]:

	Unnamed: 0	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weigh
0	False	False	False	False	False	False	False	False	False	False	Fals
1	False	False	False	False	False	False	False	False	False	False	Fals
2	False	False	False	False	False	False	True	False	False	False	Fals
3	False	False	False	False	False	False	False	False	False	False	Fals
4	False	False	False	False	False	False	False	False	False	False	Fals
...	...	...	...	...	...	...	...	...	...	...	.
1431	False	False	False	False	False	False	False	False	False	False	Fals
1432	False	False	False	False	False	False	False	False	False	False	Fals
1433	False	False	False	False	False	False	False	False	False	False	Fals
1434	False	False	False	False	True	False	False	False	False	False	Fals
1435	False	False	False	False	False	False	False	False	False	False	Fals

1436 rows × 11 columns

In [39]:
```
dataset.isnull().sum()
```

Out[39]:
```
Unnamed: 0      0
Price           0
Age             0
KM              0
FuelType      100
HP              0
MetColor      150
Automatic       0
CC              0
Doors           0
Weight          0
dtype: int64
```

4. View basic statistical details like percentile, mean, std deviation, mode, variance, skewness of Automobile data

In [50]:
```
dataset.count()
```

Out[50]:
```
Unnamed: 0    1436
Price         1436
Age           1436
KM            1436
FuelType      1336
HP            1436
MetColor      1286
Automatic     1436
CC            1436
Doors         1436
Weight        1436
dtype: int64
```

In [23]:
```
#to find the mean
print(dataset.mean())
```

```
symboling               0.834146
wheel-base             98.756585
length                174.049268
width                  65.907805
height                 53.724878
curb-weight          2555.565854
engine-size           126.907317
compression-ratio      10.142537
city-mpg               25.219512
highway-mpg            30.751220
dtype: float64

C:\Users\salon\AppData\Local\Temp\ipykernel_4484\3796844437.py:2: FutureWarni
ng: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=
None') is deprecated; in a future version this will raise TypeError.  Select
only valid columns before calling the reduction.
  print(dataset.mean())
```

In [24]:
```
#to find the standard deviation
print(dataset.std())
```

```
symboling              1.245307
wheel-base             6.021776
length                12.337289
width                  2.145204
height                 2.443522
curb-weight          520.680204
engine-size           41.642693
compression-ratio      3.972040
city-mpg               6.542142
highway-mpg            6.886443
dtype: float64

C:\Users\salon\AppData\Local\Temp\ipykernel_4484\1149059033.py:2: FutureWarni
ng: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=
None') is deprecated; in a future version this will raise TypeError.  Select
only valid columns before calling the reduction.
  print(dataset.std())
```

In [25]:
```
#to find the mode
print(dataset.mode())
```

```
   symboling normalized-losses    make fuel-type aspiration num-of-doors  \
0        0.0                 ?  toyota       gas        std         four
1        NaN               NaN     NaN       NaN        NaN          NaN

  body-style drive-wheels engine-location  wheel-base  ...  engine-size  \
0      sedan          fwd           front        94.5  ...           92
1        NaN          NaN             NaN         NaN  ...          122

   fuel-system  bore  stroke compression-ratio horsepower  peak-rpm city-mpg
\
0         mpfi  3.62     3.4              9.0         68      5500     31.0
1          NaN   NaN     NaN              NaN        NaN       NaN      NaN

  highway-mpg price
0        25.0     ?
1         NaN   NaN

[2 rows x 26 columns]
```

In [26]:
```
#to find the variance
print(dataset.var())
```

```
symboling               1.550789
wheel-base             36.261782
length                152.208688
width                   4.601900
height                  5.970800
curb-weight        271107.874319
engine-size          1734.113917
compression-ratio      15.777104
city-mpg               42.799617
highway-mpg            47.423099
dtype: float64

C:\Users\salon\AppData\Local\Temp\ipykernel_4484\1526402557.py:2: FutureWarni
ng: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=
None') is deprecated; in a future version this will raise TypeError.  Select
only valid columns before calling the reduction.
  print(dataset.var())
```

In [27]:
```
#to find the skewness of data
print(dataset.skew())
```

```
symboling           0.211072
wheel-base          1.050214
length              0.155954
width               0.904003
height              0.063123
curb-weight         0.681398
engine-size         1.947655
compression-ratio   2.610862
city-mpg            0.663704
highway-mpg         0.539997
dtype: float64

C:\Users\salon\AppData\Local\Temp\ipykernel_4484\2158001515.py:2: FutureWarni
ng: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=
None') is deprecated; in a future version this will raise TypeError.  Select
only valid columns before calling the reduction.
  print(dataset.skew())
```

5. Find the most expensive car company name.

In [39]:
```
df1=dataset[['price']][df.price==df['price'].max()]
df1
```

Out[39]:

	price
9	?
44	?
45	?
129	?

6. Print all Toyota cars available.

In [34]:
```
all_toyota=dataset.groupby('make')
toyotaDf=all_toyota.get_group('toyota')
toyotaDf
```

Out[34]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	whee bas
150	1	87	toyota	gas	std	two	hatchback	fwd	front	95
151	1	87	toyota	gas	std	two	hatchback	fwd	front	95
152	1	74	toyota	gas	std	four	hatchback	fwd	front	95
153	0	77	toyota	gas	std	four	wagon	fwd	front	95
154	0	81	toyota	gas	std	four	wagon	4wd	front	95
155	0	91	toyota	gas	std	four	wagon	4wd	front	95
156	0	91	toyota	gas	std	four	sedan	fwd	front	95
157	0	91	toyota	gas	std	four	hatchback	fwd	front	95
158	0	91	toyota	diesel	std	four	sedan	fwd	front	95
159	0	91	toyota	diesel	std	four	hatchback	fwd	front	95
160	0	91	toyota	gas	std	four	sedan	fwd	front	95
161	0	91	toyota	gas	std	four	hatchback	fwd	front	95
162	0	91	toyota	gas	std	four	sedan	fwd	front	95
163	1	168	toyota	gas	std	two	sedan	rwd	front	94
164	1	168	toyota	gas	std	two	hatchback	rwd	front	94
165	1	168	toyota	gas	std	two	sedan	rwd	front	94
166	1	168	toyota	gas	std	two	hatchback	rwd	front	94
167	2	134	toyota	gas	std	two	hardtop	rwd	front	98
168	2	134	toyota	gas	std	two	hardtop	rwd	front	98
169	2	134	toyota	gas	std	two	hatchback	rwd	front	98
170	2	134	toyota	gas	std	two	hardtop	rwd	front	98
171	2	134	toyota	gas	std	two	hatchback	rwd	front	98
172	2	134	toyota	gas	std	two	convertible	rwd	front	98
173	-1	65	toyota	gas	std	four	sedan	fwd	front	102
174	-1	65	toyota	diesel	turbo	four	sedan	fwd	front	102
175	-1	65	toyota	gas	std	four	hatchback	fwd	front	102
176	-1	65	toyota	gas	std	four	sedan	fwd	front	102
177	-1	65	toyota	gas	std	four	hatchback	fwd	front	102
178	3	197	toyota	gas	std	two	hatchback	rwd	front	102
179	3	197	toyota	gas	std	two	hatchback	rwd	front	102
180	-1	90	toyota	gas	std	four	sedan	rwd	front	104

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	whee bas

7. Find for each company's highest price car

In [37]:
```
dataset.groupby('make')
dataset[['make','price']][df.price==df['price'].max()]
```

Out[37]:

	make	price
9	audi	?
44	isuzu	?
45	isuzu	?
129	porsche	?

8. Find the average mileage of each car making company.

In [40]:
```
avg_mileage=df.groupby('make')
avg_mileage=avg_mileage['make','highway-mpg'].mean()
avg_mileage
```

```
C:\Users\salon\AppData\Local\Temp\ipykernel_4484\1284515093.py:2: FutureWarni
ng: Indexing with multiple keys (implicitly converted to a tuple of keys) wil
l be deprecated, use a list instead.
  avg_mileage=avg_mileage['make','highway-mpg'].mean()
```

Out[40]:

make	highway-mpg
alfa-romero	26.666667
audi	24.142857
bmw	25.375000
chevrolet	46.333333
dodge	34.111111
honda	35.461538
isuzu	36.000000
jaguar	18.333333
mazda	31.941176
mercedes-benz	21.000000
mercury	24.000000
mitsubishi	31.153846
nissan	32.944444
peugot	26.636364
plymouth	34.142857
porsche	26.000000
renault	31.000000
saab	27.333333
subaru	30.750000
toyota	32.906250
volkswagen	34.916667
volvo	25.818182

9. Sort all cars by price column.

In [41]:
```
df=dataset.sort_values(by=['price'])
df
```

Out[41]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	w
147	0	89	subaru	gas	std	four	wagon	fwd	front	
62	0	115	mazda	gas	std	four	sedan	fwd	front	
40	0	85	honda	gas	std	four	sedan	fwd	front	
42	1	107	honda	gas	std	two	sedan	fwd	front	
61	1	129	mazda	gas	std	two	hatchback	fwd	front	
...	...	...	...	...	...	...	...	...	...	
188	2	94	volkswagen	gas	std	four	sedan	fwd	front	
45	0	?	isuzu	gas	std	four	sedan	fwd	front	
44	1	?	isuzu	gas	std	two	sedan	fwd	front	
129	1	?	porsche	gas	std	two	hatchback	rwd	front	
9	0	?	audi	gas	turbo	two	hatchback	4wd	front	

205 rows × 26 columns

10. Create the horizontal bar plot for Automobile dataset

In [43]:
```
df1=pd.DataFrame(np.random.rand(10,3),columns=['a','b','c'])
print(df1)
```

```
          a         b         c
0  0.765909  0.507744  0.904466
1  0.247291  0.810380  0.033680
2  0.662060  0.415940  0.050235
3  0.930562  0.572470  0.629153
4  0.241792  0.960058  0.638245
5  0.933473  0.864184  0.251297
6  0.434384  0.635849  0.914356
7  0.335788  0.011190  0.076124
8  0.256613  0.083312  0.777009
9  0.437438  0.132157  0.277054
```

In [48]:
```
df1.plot.barh(stacked=False)
```

Out[48]:
```
<AxesSubplot:>
```

11. Create a pair plot.

In [47]:
```
sns.pairplot(dataset)
```

Out[47]:
```
<seaborn.axisgrid.PairGrid at 0x277206a1a30>
```

12. Print correlation matrix for this Automobile dataset and show heatmap.

In [53]:
```
corr_matrix=df.corr()
print(corr_matrix)
```

```
            Unnamed: 0     Price       Age  MetColor  Automatic        CC  \
Unnamed: 0    1.000000 -0.738289  0.907090 -0.069409   0.066299 -0.184490
Price        -0.738289  1.000000 -0.878407  0.100920   0.033081  0.165067
Age           0.907090 -0.878407  1.000000 -0.088232   0.032573 -0.120706
MetColor     -0.069409  0.100920 -0.088232  1.000000  -0.011450  0.032108
Automatic     0.066299  0.033081  0.032573 -0.011450   1.000000 -0.069321
CC           -0.184490  0.165067 -0.120706  0.032108  -0.069321  1.000000
Weight       -0.414577  0.581198 -0.464299  0.046614   0.057249  0.651450

              Weight
Unnamed: 0 -0.414577
Price       0.581198
Age        -0.464299
MetColor    0.046614
Automatic   0.057249
CC          0.651450
Weight      1.000000
```

In [54]:
```
sns.heatmap(corr_matrix,annot=True)
```

Out[54]:
```
<AxesSubplot:>
```

In [ ]:
```

```