Data Acquisition

There are various formats for a dataset such as .csv, .json, .xlsx etc. The dataset can be stored in different places on your local machine or sometimes online.

```
import pandas as pd

df = pd.read_csv('dataset/uncleaned_auto_data.csv')
print(" 5 rows of the dataframe")
df.head(5)
```

```
 5 rows of the dataframe
```

	Unnamed: 0.1	Unnamed: 0	symboling	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels	engine-location	...	compression-ratio
0	0	0	3	122	alfa-romero	std	two	convertible	rwd	front	...	9.0
1	1	1	3	122	alfa-romero	std	two	convertible	rwd	front	...	9.0
2	2	2	1	122	alfa-romero	std	two	hatchback	rwd	front	...	9.0
3	3	3	2	164	audi	std	four	sedan	fwd	front	...	10.0
4	4	4	2	164	audi	std	four	sedan	4wd	front	...	8.0

5 rows × 31 columns

```
   Unnamed: 0.1  Unnamed: 0  symboling  normalized-losses         make  \
0             0           0          3                122  alfa-romero   
1             1           1          3                122  alfa-romero   
2             2           2          1                122  alfa-romero   
3             3           3          2                164         audi   
4             4           4          2                164         audi   

  aspiration num-of-doors   body-style drive-wheels engine-location  ...  \
0        std          two  convertible          rwd           front  ...   
1        std          two  convertible          rwd           front  ...   
2        std          two    hatchback          rwd           front  ...   
3        std         four        sedan          fwd           front  ...   
4        std         four        sedan          4wd           front  ...   

   compression-ratio  horsepower  peak-rpm  city-mpg  highway-mpg    price  \
0                9.0       111.0    5000.0        21           27  13495.0   
1                9.0       111.0    5000.0        21           27  16500.0   
2                9.0       154.0    5000.0        19           26  16500.0   
```

```
3                 10.0        102.0     5500.0         24             30  13950.0
4                  8.0        115.0     5500.0         18             22  17450.0
```

Drop the missing values along the column "price"

```
df2 = df.drop(columns=['Unnamed: 0.1', 'Unnamed: 0'], axis=0)
```

```
print("COLUMN NAMES")
print(df2.columns)
```

```
COLUMN NAMES
Index(['symboling', 'normalized-losses', 'make', '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',
       'city-L/100km', 'horsepower-binned', 'diesel', 'gas'],
      dtype='object')
```

```
df2.to_csv('dataset/final_data.csv' , index=False)
```

Data Types

Data has a variety of types.The main types stored in Pandas dataframes are **object**, **float**, **int**, **bool** and **datetime64**. In order to better learn about each attribute, it is always good to know the data type of each column. In Pandas:

```
df2.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
```

It returns a Series with the data type of each column.

It is clear to see that the data type of "symboling" and "curb-weight" are `int64`, "normalized-losses" is `object`, and "wheel-base" is `float64`, etc.

```
df2.describe()
```

	Unnamed: 0.1	Unnamed: 0	symboling	normalized-losses	wheel-base	length	width	height	curb-weight	engi
count	201.000000	201.000000	201.000000	201.00000	201.000000	201.000000	201.000000	201.000000	201.000000	201.
mean	100.000000	100.000000	0.840796	122.00000	98.797015	0.837102	0.915126	53.766667	2555.666667	126.8
std	58.167861	58.167861	1.254802	31.99625	6.066366	0.059213	0.029187	2.447822	517.296727	41.5
min	0.000000	0.000000	-2.000000	65.00000	86.600000	0.678039	0.837500	47.800000	1488.000000	61.0
25%	50.000000	50.000000	0.000000	101.00000	94.500000	0.801538	0.890278	52.000000	2169.000000	98.0
50%	100.000000	100.000000	1.000000	122.00000	97.000000	0.832292	0.909722	54.100000	2414.000000	120.
75%	150.000000	150.000000	2.000000	137.00000	102.400000	0.881788	0.925000	55.500000	2926.000000	141.
max	200.000000	200.000000	3.000000	256.00000	120.900000	1.000000	1.000000	59.800000	4066.000000	326.

8 rows × 21 columns

```
       Unnamed: 0.1  Unnamed: 0   symboling  normalized-losses  wheel-base  \
count    201.000000  201.000000  201.000000          201.00000  201.000000   
mean     100.000000  100.000000    0.840796          122.00000   98.797015   
std       58.167861   58.167861    1.254802           31.99625    6.066366   
min        0.000000    0.000000   -2.000000           65.00000   86.600000   
25%       50.000000   50.000000    0.000000          101.00000   94.500000   
50%      100.000000  100.000000    1.000000          122.00000   97.000000   
75%      150.000000  150.000000    2.000000          137.00000  102.400000   
max      200.000000  200.000000    3.000000          256.00000  120.900000   

           length       width      height  curb-weight  engine-size  ...  \
count  201.000000  201.000000  201.000000   201.000000   201.000000  ...   
mean     0.837102    0.915126   53.766667  2555.666667   126.875622  ...   
std      0.059213    0.029187    2.447822   517.296727    41.546834  ...   
min      0.678039    0.837500   47.800000  1488.000000    61.000000  ...   
25%      0.801538    0.890278   52.000000  2169.000000    98.000000  ...   
50%      0.832292    0.909722   54.100000  2414.000000   120.000000  ...   
75%      0.881788    0.925000   55.500000  2926.000000   141.000000  ...   
max      1.000000    1.000000   59.800000  4066.000000   326.000000  ...   
```

This shows the statistical summary of all numeric-typed (int, float) columns.
For example, the attribute "symboling" has 205 counts, the mean value of this column is 0.83, the standard deviation is 1.25, the minimum value is -2, 25th percentile is 0, 50th percentile is 1, 75th percentile is 2, and the maximum value is 3.

```
# describe all the columns in "df"
df2.describe(include = "all")
```

	symboling	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	...	compressi ratio
count	201.000000	201.00000	201	201	201	201	201	201	201.000000	201.000000	...	201.00000
unique	NaN	NaN	22	2	2	5	3	2	NaN	NaN	...	NaN
top	NaN	NaN	toyota	std	four	sedan	fwd	front	NaN	NaN	...	NaN
freq	NaN	NaN	32	165	115	94	118	198	NaN	NaN	...	NaN
mean	0.840796	122.00000	NaN	NaN	NaN	NaN	NaN	NaN	98.797015	0.837102	...	10.164279
std	1.254802	31.99625	NaN	NaN	NaN	NaN	NaN	NaN	6.066366	0.059213	...	4.004965
min	-2.000000	65.00000	NaN	NaN	NaN	NaN	NaN	NaN	86.600000	0.678039	...	7.000000
25%	0.000000	101.00000	NaN	NaN	NaN	NaN	NaN	NaN	94.500000	0.801538	...	8.600000
50%	1.000000	122.00000	NaN	NaN	NaN	NaN	NaN	NaN	97.000000	0.832292	...	9.000000
75%	2.000000	137.00000	NaN	NaN	NaN	NaN	NaN	NaN	102.400000	0.881788	...	9.400000
max	3.000000	256.00000	NaN	NaN	NaN	NaN	NaN	NaN	120.900000	1.000000	...	23.000000

11 rows × 29 columns

```
        symboling  normalized-losses    make aspiration num-of-doors  \
count   201.000000          201.00000     201        201          201
unique         NaN                NaN      22          2            2
top            NaN                NaN  toyota        std         four
freq           NaN                NaN      32        165          115
mean      0.840796          122.00000     NaN        NaN          NaN
std       1.254802           31.99625     NaN        NaN          NaN
min      -2.000000           65.00000     NaN        NaN          NaN
25%       0.000000          101.00000     NaN        NaN          NaN
50%       1.000000          122.00000     NaN        NaN          NaN
75%       2.000000          137.00000     NaN        NaN          NaN
max       3.000000          256.00000     NaN        NaN          NaN

       body-style drive-wheels engine-location  wheel-base      length  ...  \
count         201          201             201  201.000000  201.000000  ...
unique          5            3               2         NaN         NaN  ...
top         sedan          fwd           front         NaN         NaN  ...
freq           94          118             198         NaN         NaN  ...
mean          NaN          NaN             NaN   98.797015    0.837102  ...
std           NaN          NaN             NaN    6.066366    0.059213  ...
```

Now, it provides the statistical summary of all the columns, including object-typed attributes.
We can now see how many unique values, which is the top value and the frequency of top value in the object-typed columns.
Some values in the table above show as "NaN", this is because those numbers are not available regarding a particular column type.

```python
df2[['length','compression-ratio']].describe()
```

	length	compression-ratio
count	201.000000	201.000000
mean	0.837102	10.164279
std	0.059213	4.004965
min	0.678039	7.000000
25%	0.801538	8.600000
50%	0.832292	9.000000
75%	0.881788	9.400000
max	1.000000	23.000000

```
           length  compression-ratio
count  201.000000         201.000000
mean     0.837102          10.164279
std      0.059213           4.004965
min      0.678039           7.000000
25%      0.801538           8.600000
50%      0.832292           9.000000
75%      0.881788           9.400000
max      1.000000          23.000000
```

```python
df2.info
```

```
<bound method DataFrame.info of      symboling  normalized-losses         make aspirati
0            3                 122  alfa-romero        std          two
1            3                 122  alfa-romero        std          two
2            1                 122  alfa-romero        std          two
3            2                 164         audi        std         four
4            2                 164         audi        std         four
..         ...                 ...          ...        ...          ...
196         -1                  95        volvo        std         four
197         -1                  95        volvo      turbo         four
198         -1                  95        volvo        std         four
199         -1                  95        volvo      turbo         four
200         -1                  95        volvo      turbo         four

      body-style drive-wheels engine-location  wheel-base    length  ...  \
0    convertible          rwd           front        88.6  0.811148  ...
1    convertible          rwd           front        88.6  0.811148  ...
2      hatchback          rwd           front        94.5  0.822681  ...
3          sedan          fwd           front        99.8  0.848630  ...
4          sedan          4wd           front        99.4  0.848630  ...
```

It shows us that the whole data frame has 205 rows and 26 columns in total.