Data Analysis with Python

Introduction

Welcome!

In this section, you will learn how to approach data acquisition in various ways, and obtain necessary insights from a dataset. By the end of this lab, you will successfully load the data into Jupyter Notebook, and gain some fundamental insights via Pandas Library.

Table of Contents

Estimated Time Needed: **10 min**

Data Acquisition

There are various formats for a dataset, .csv, .json, .xlsx etc. The dataset can be stored in different places, on your local machine or sometimes online.
In this section, you will learn how to load a dataset into our Jupyter Notebook.
In our case, the Automobile Dataset is an online source, and it is in CSV (comma separated value) format. Let's use this dataset as an example to practice data reading.

- data source: https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.data
- data type: csv

The Pandas Library is a useful tool that enables us to read various datasets into a data frame; our Jupyter notebook platforms have a built-in **Pandas Library** so that all we need to do is import Pandas without installing.

[1]:
```
# import pandas library
import pandas as pd
```

Read Data

We use `pandas.read_csv()` function to read the csv file. In the bracket, we put the file path along with a quotation mark, so that pandas will read the file into a data frame from that address. The file path can be either an URL or your local file address.
Because the data does not include headers, we can add an argument `headers = None` inside the `read_csv()` method, so that pandas will not automatically set the first row as a header.
You can also assign the dataset to any variable you create.

This dataset was hosted on IBM Cloud object click HERE for free storage.

[2]:
```
# Import pandas library
import pandas as pd

# Read the online file by the URL provides above, and assign it to variable "df"
other_path = "https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/DA0101EN/auto.csv"
df = pd.read_csv(other_path, header=None)
```

After reading the dataset, we can use the `dataframe.head(n)` method to check the top n rows of the dataframe; where n is an integer. Contrary to `dataframe.head(n)`, `dataframe.tail(n)` will show you the bottom n rows of the dataframe.

[3]:
```
# show the first 5 rows using dataframe.head() method
print("The first 5 rows of the dataframe")
df.head(5)
```

```
The first 5 rows of the dataframe
```

[3]:

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21	22	23	24	25
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500	24	30	13950
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500	18	22	17450

5 rows × 26 columns

Question #1:

check the bottom 10 rows of data frame "df".

[4]:
```
# Write your code below and press Shift+Enter to execute
df.tail(10)
```

[4]:

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21	22	23	24	25
195	-1	74	volvo	gas	std	four	wagon	rwd	front	104.3	...	141	mpfi	3.78	3.15	9.5	114	5400	23	28	13415
196	-2	103	volvo	gas	std	four	sedan	rwd	front	104.3	...	141	mpfi	3.78	3.15	9.5	114	5400	24	28	15985

197	-1	74	volvo	gas	std	four	wagon	rwd	front	104.3	...	141	mpfi	3.78	3.15	9.5	114	5400	24	28	16515
198	-2	103	volvo	gas	turbo	four	sedan	rwd	front	104.3	...	130	mpfi	3.62	3.15	7.5	162	5100	17	22	18420
199	-1	74	volvo	gas	turbo	four	wagon	rwd	front	104.3	...	130	mpfi	3.62	3.15	7.5	162	5100	17	22	18950
200	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114	5400	23	28	16845
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	8.7	160	5300	19	25	19045
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	173	mpfi	3.58	2.87	8.8	134	5500	18	23	21485
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1	...	145	idi	3.01	3.40	23.0	106	4800	26	27	22470
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114	5400	19	25	22625

10 rows × 26 columns

Question #1 Answer:

Run the code below for the solution!

Double-click **here** for the solution.

Add Headers

Take a look at our dataset; pandas automatically set the header by an integer from 0.

To better describe our data we can introduce a header, this information is available at: https://archive.ics.uci.edu/ml/datasets/Automobile

Thus, we have to add headers manually.

Firstly, we create a list "headers" that include all column names in order. Then, we use `dataframe.columns = headers` to replace the headers by the list we created.

[5]:
```
# create headers list
headers = ["symboling","normalized-losses","make","fuel-type","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"]
print("headers\n", headers)
```

```
headers
 ['symboling', 'normalized-losses', 'make', 'fuel-type', 'aspiration', 'num-of-doors', 'body-style', 'drive-wheels', 'engine-location', 'wheel-base', 'length', 'width', 'height', 'curb-weight', 'engine-ty
pe', 'num-of-cylinders', 'engine-size', 'fuel-system', 'bore', 'stroke', 'compression-ratio', 'horsepower', 'peak-rpm', 'city-mpg', 'highway-mpg', 'price']
```

We replace headers and recheck our data frame

[9]:
```
df.columns = headers
df.head(10)
```

[9]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500	24	30	13950
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500	18	22	17450
5	2	?	audi	gas	std	two	sedan	fwd	front	99.8	...	136	mpfi	3.19	3.40	8.5	110	5500	19	25	15250
6	1	158	audi	gas	std	four	sedan	fwd	front	105.8	...	136	mpfi	3.19	3.40	8.5	110	5500	19	25	17710
7	1	?	audi	gas	std	four	wagon	fwd	front	105.8	...	136	mpfi	3.19	3.40	8.5	110	5500	19	25	18920
8	1	158	audi	gas	turbo	four	sedan	fwd	front	105.8	...	131	mpfi	3.13	3.40	8.3	140	5500	17	20	23875
9	0	?	audi	gas	turbo	two	hatchback	4wd	front	99.5	...	131	mpfi	3.13	3.40	7.0	160	5500	16	22	?

10 rows × 26 columns

we can drop missing values along the column "price" as follows

[10]:
```
df.dropna(subset=["price"], axis=0)
```

[10]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.00	111	5000	21	27	13495
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.00	111	5000	21	27	16500
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.00	154	5000	19	26	16500
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.00	102	5500	24	30	13950
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.00	115	5500	18	22	17450
5	2	?	audi	gas	std	two	sedan	fwd	front	99.8	...	136	mpfi	3.19	3.40	8.50	110	5500	19	25	15250
6	1	158	audi	gas	std	four	sedan	fwd	front	105.8	...	136	mpfi	3.19	3.40	8.50	110	5500	19	25	17710
7	1	?	audi	gas	std	four	wagon	fwd	front	105.8	...	136	mpfi	3.19	3.40	8.50	110	5500	19	25	18920
8	1	158	audi	gas	turbo	four	sedan	fwd	front	105.8	...	131	mpfi	3.13	3.40	8.30	140	5500	17	20	23875
9	0	?	audi	gas	turbo	two	hatchback	4wd	front	99.5	...	131	mpfi	3.13	3.40	7.00	160	5500	16	22	?
10	2	192	bmw	gas	std	two	sedan	rwd	front	101.2	...	108	mpfi	3.50	2.80	8.80	101	5800	23	29	16430
11	0	192	bmw	gas	std	four	sedan	rwd	front	101.2	...	108	mpfi	3.50	2.80	8.80	101	5800	23	29	16925
12	0	188	bmw	gas	std	two	sedan	rwd	front	101.2	...	164	mpfi	3.31	3.19	9.00	121	4250	21	28	20970
13	0	188	bmw	gas	std	four	sedan	rwd	front	101.2	...	164	mpfi	3.31	3.19	9.00	121	4250	21	28	21105
14	1	?	bmw	gas	std	four	sedan	rwd	front	103.5	...	164	mpfi	3.31	3.19	9.00	121	4250	20	25	24565
15	0	?	bmw	gas	std	four	sedan	rwd	front	103.5	...	209	mpfi	3.62	3.39	8.00	182	5400	16	22	30760
16	0	?	bmw	gas	std	two	sedan	rwd	front	103.5	...	209	mpfi	3.62	3.39	8.00	182	5400	16	22	41315
17	0	?	bmw	gas	std	four	sedan	rwd	front	110.0	...	209	mpfi	3.62	3.39	8.00	182	5400	15	20	36880
18	2	121	chevrolet	gas	std	two	hatchback	fwd	front	88.4	...	61	2bbl	2.91	3.03	9.50	48	5100	47	53	5151
19	1	98	chevrolet	gas	std	two	hatchback	fwd	front	94.5	...	90	2bbl	3.03	3.11	9.60	70	5400	38	43	6295
20	0	81	chevrolet	gas	std	four	sedan	fwd	front	94.5	...	90	2bbl	3.03	3.11	9.60	70	5400	38	43	6575
21	1	118	dodge	gas	std	two	hatchback	fwd	front	93.7	...	90	2bbl	2.97	3.23	9.41	68	5500	37	41	5572
22	1	118	dodge	gas	std	two	hatchback	fwd	front	93.7	...	90	2bbl	2.97	3.23	9.40	68	5500	31	38	6377
23	1	118	dodge	gas	turbo	two	hatchback	fwd	front	93.7	...	98	mpfi	3.03	3.39	7.60	102	5500	24	30	7957
24	1	148	dodge	gas	std	four	hatchback	fwd	front	93.7	...	90	2bbl	2.97	3.23	9.40	68	5500	31	38	6229
25	1	148	dodge	gas	std	four	sedan	fwd	front	93.7	...	90	2bbl	2.97	3.23	9.40	68	5500	31	38	6692
26	1	148	dodge	gas	std	four	sedan	fwd	front	93.7	...	90	2bbl	2.97	3.23	9.40	68	5500	31	38	7609
27	1	148	dodge	gas	turbo	?	sedan	fwd	front	93.7	...	98	mpfi	3.03	3.39	7.60	102	5500	24	30	8558
28	-1	110	dodge	gas	std	four	wagon	fwd	front	103.3	...	122	2bbl	3.34	3.46	8.50	88	5000	24	30	8921
29	3	145	dodge	gas	turbo	two	hatchback	fwd	front	95.9	...	156	mfi	3.60	3.90	7.00	145	5000	19	24	12964
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
175	-1	65	toyota	gas	std	four	hatchback	fwd	front	102.4	...	122	mpfi	3.31	3.54	8.70	92	4200	27	32	9988
176	-1	65	toyota	gas	std	four	sedan	fwd	front	102.4	...	122	mpfi	3.31	3.54	8.70	92	4200	27	32	10898
177	-1	65	toyota	gas	std	four	hatchback	fwd	front	102.4	...	122	mpfi	3.31	3.54	8.70	92	4200	27	32	11248
178	3	197	toyota	gas	std	two	hatchback	rwd	front	102.9	...	171	mpfi	3.27	3.35	9.30	161	5200	20	24	16558
179	3	197	toyota	gas	std	two	hatchback	rwd	front	102.9	...	171	mpfi	3.27	3.35	9.30	161	5200	19	24	15998

180	-1	90	toyota	gas	std	four	sedan	rwd	front	104.5	...	171	mpfi	3.27	3.35	9.20	156	5200	20	24	15690
181	-1	?	toyota	gas	std	four	wagon	rwd	front	104.5	...	161	mpfi	3.27	3.35	9.20	156	5200	19	24	15750
182	2	122	volkswagen	diesel	std	two	sedan	fwd	front	97.3	...	97	idi	3.01	3.40	23.00	52	4800	37	46	7775
183	2	122	volkswagen	gas	std	two	sedan	fwd	front	97.3	...	109	mpfi	3.19	3.40	9.00	85	5250	27	34	7975
184	2	94	volkswagen	diesel	std	four	sedan	fwd	front	97.3	...	97	idi	3.01	3.40	23.00	52	4800	37	46	7995
185	2	94	volkswagen	gas	std	four	sedan	fwd	front	97.3	...	109	mpfi	3.19	3.40	9.00	85	5250	27	34	8195
186	2	94	volkswagen	gas	std	four	sedan	fwd	front	97.3	...	109	mpfi	3.19	3.40	9.00	85	5250	27	34	8495
187	2	94	volkswagen	diesel	turbo	four	sedan	fwd	front	97.3	...	97	idi	3.01	3.40	23.00	68	4500	37	42	9495
188	2	94	volkswagen	gas	std	four	sedan	fwd	front	97.3	...	109	mpfi	3.19	3.40	10.00	100	5500	26	32	9995
189	3	?	volkswagen	gas	std	two	convertible	fwd	front	94.5	...	109	mpfi	3.19	3.40	8.50	90	5500	24	29	11595
190	3	256	volkswagen	gas	std	two	hatchback	fwd	front	94.5	...	109	mpfi	3.19	3.40	8.50	90	5500	24	29	9980
191	0	?	volkswagen	gas	std	four	sedan	fwd	front	100.4	...	136	mpfi	3.19	3.40	8.50	110	5500	19	24	13295
192	0	?	volkswagen	diesel	turbo	four	sedan	fwd	front	100.4	...	97	idi	3.01	3.40	23.00	68	4500	33	38	13845
193	0	?	volkswagen	gas	std	four	wagon	fwd	front	100.4	...	109	mpfi	3.19	3.40	9.00	88	5500	25	31	12290
194	-2	103	volvo	gas	std	four	sedan	rwd	front	104.3	...	141	mpfi	3.78	3.15	9.50	114	5400	23	28	12940
195	-1	74	volvo	gas	std	four	wagon	rwd	front	104.3	...	141	mpfi	3.78	3.15	9.50	114	5400	23	28	13415
196	-2	103	volvo	gas	std	four	sedan	rwd	front	104.3	...	141	mpfi	3.78	3.15	9.50	114	5400	24	28	15985
197	-1	74	volvo	gas	std	four	wagon	rwd	front	104.3	...	141	mpfi	3.78	3.15	9.50	114	5400	24	28	16515
198	-2	103	volvo	gas	turbo	four	sedan	rwd	front	104.3	...	130	mpfi	3.62	3.15	7.50	162	5100	17	22	18420
199	-1	74	volvo	gas	turbo	four	wagon	rwd	front	104.3	...	130	mpfi	3.62	3.15	7.50	162	5100	17	22	18950
200	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.50	114	5400	23	28	16845
201	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	8.70	160	5300	19	25	19045
202	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	173	mpfi	3.58	2.87	8.80	134	5500	18	23	21485
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1	...	145	idi	3.01	3.40	23.00	106	4800	26	27	22470
204	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.50	114	5400	19	25	22625

205 rows × 26 columns

Now, we have successfully read the raw dataset and add the correct headers into the data frame.

Question #2:

Find the name of the columns of the dataframe

[13]:
```
# Write your code below and press Shift+Enter to execute
df.columns
```

[13]:
```
Index(['symboling', 'normalized-losses', 'make', 'fuel-type', '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'],
      dtype='object')
```

•••

•••

```
df.to_csv("automobile.csv", index=False)
```

We can also read and save other file formats. we can use similar functions to `pd.read_csv()` and `df.to_csv()` for other data formats, the functions are listed in the following table:

Read/Save Other Data Formats

Data Formate	Read	Save
csv	`pd.read_csv()`	`df.to_csv()`
json	`pd.read_json()`	`df.to_json()`
excel	`pd.read_excel()`	`df.to_excel()`
hdf	`pd.read_hdf()`	`df.to_hdf()`
sql	`pd.read_sql()`	`df.to_sql()`
...	...	...

Basic Insight of Dataset

After reading data into Pandas dataframe, it is time for us to explore the dataset.
There are several ways to obtain essential insights of the data to help us better understand our dataset.

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 for us to know the data type of each column. In Pandas:

[14]:
```
df.dtypes
```

[14]:
```
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
```

returns a Series with the data type of each column.

[15]:
```
# check the data type of data frame "df" by .dtypes
print(df.dtypes)
```

```
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
```

As a result, as shown above, 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.

These data types can be changed; we will learn how to accomplish this in a later module.

Describe

If we would like to get a statistical summary of each column, such as count, column mean value, column standard deviation, etc. We use the describe method:

```
dataframe.describe()
```

This method will provide various summary statistics, excluding `NaN` (Not a Number) values.

[16]:
```
df.describe()
```

[16]:

	symboling	wheel-base	length	width	height	curb-weight	engine-size	compression-ratio	city-mpg	highway-mpg
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	10.142537	25.219512	30.751220
std	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	3.972040	6.542142	6.886443
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	7.000000	13.000000	16.000000
25%	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	8.600000	19.000000	25.000000
50%	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	9.000000	24.000000	30.000000
75%	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	9.400000	30.000000	34.000000
max	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	23.000000	49.000000	54.000000

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.
However, what if we would also like to check all the columns including those that are of type object.

You can add an argument `include = "all"` inside the bracket. Let's try it again.

[17]:
```
# describe all the columns in "df"
df.describe(include = "all")
```

[17]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price
count	205.000000	205	205	205	205	205	205	205	205	205.000000	...	205.000000	205	205	205	205.000000	205	205	205.000000	205.000000	205
unique	NaN	52	22	2	2	3	5	3	2	NaN	...	NaN	8	39	37	NaN	60	24	NaN	NaN	187
top	NaN	?	toyota	gas	std	four	sedan	fwd	front	NaN	...	NaN	mpfi	3.62	3.40	NaN	68	5500	NaN	NaN	?
freq	NaN	41	32	185	168	114	96	120	202	NaN	...	NaN	94	23	20	NaN	19	37	NaN	NaN	4
mean	0.834146	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	98.756585	...	126.907317	NaN	NaN	NaN	10.142537	NaN	NaN	25.219512	30.751220	NaN
std	1.245307	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	6.021776	...	41.642693	NaN	NaN	NaN	3.972040	NaN	NaN	6.542142	6.886443	NaN
min	-2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	86.600000	...	61.000000	NaN	NaN	NaN	7.000000	NaN	NaN	13.000000	16.000000	NaN
25%	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	94.500000	...	97.000000	NaN	NaN	NaN	8.600000	NaN	NaN	19.000000	25.000000	NaN
50%	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	97.000000	...	120.000000	NaN	NaN	NaN	9.000000	NaN	NaN	24.000000	30.000000	NaN
75%	2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	102.400000	...	141.000000	NaN	NaN	NaN	9.400000	NaN	NaN	30.000000	34.000000	NaN
max	3.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	120.900000	...	326.000000	NaN	NaN	NaN	23.000000	NaN	NaN	49.000000	54.000000	NaN

11 rows × 26 columns

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.

Question #3:

You can select the columns of a data frame by indicating the name of each column, for example, you can select the three columns as follows:

```
dataframe[[' column 1 ',column 2', 'column 3']]
```

Where "column" is the name of the column, you can apply the method ".describe()" to get the statistics of those columns as follows:

```
dataframe[[' column 1 ',column 2', 'column 3'] ].describe()
```

Apply the method to ".describe()" to the columns 'length' and 'compression-ratio'.

[18]:
```
# Write your c'ode below and press Shift+Enter to execute
df[['length', 'compression-ratio']].describe()
```

[18]:

	length	compression-ratio
count	205.000000	205.000000
mean	174.049268	10.142537
std	12.337289	3.972040
min	141.100000	7.000000
25%	166.300000	8.600000
50%	173.200000	9.000000
75%	183.100000	9.400000
max	208.100000	23.000000

Double-click **here** for the solution.

Info

Another method you can use to check your dataset is:

```
dataframe.info
```

It provide a concise summary of your DataFrame.

[19]:
```
# look at the info of "df"
df.info
```

[19]:
```
<bound method DataFrame.info of      symboling normalized-losses         make 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    3.40             10.0        102
4            136         mpfi  3.19    3.40              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    3.40             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]>
```

Here we are able to see the information of our dataframe, with the top 30 rows and the bottom 30 rows.

And, it also shows us the whole data frame has 205 rows and 26 columns in total.

Excellent! You have just completed the Introduction Notebook!

```
<p><a href="https://cocl.us/corsera_da0101en_notebook_bottom"><img src="https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/DA0101EN/Images/BottomAd.png"
width="750" align="center"></a></p>
```

About the Authors:

This notebook was written by Mahdi Noorian PhD, Joseph Santarcangelo, Bahare Talayian, Eric Xiao, Steven Dong, Parizad, Hima Vsudevan and Fiorella Wenver and Yi Yao.

Joseph Santarcangelo is a Data Scientist at IBM, and holds a PhD in Electrical Engineering. His research focused on using Machine Learning, Signal Processing, and Computer Vision to determine how videos impact human cognition. Joseph has been working for IBM since he completed his PhD.
