Introduction Notebook

Estimated time needed: **10** minutes

Objectives

After completing this lab you will be able to:

- Acquire data in various ways
- Obtain insights from data with Pandas library

you are running the lab in your browser, so we will install the libraries using `piplite`

```
#you are running the lab in your  browser, so we will install the
libraries using ``piplite``
import piplite
import micropip
await piplite.install(['pandas'])
await piplite.install(['matplotlib'])
await piplite.install(['scipy'])
await piplite.install(['seaborn'])
await micropip.install(['ipywidgets'],keep_going=True)
await micropip.install(['tqdm'],keep_going=True)
```

If you run the lab locally using Anaconda, you can load the correct library and versions by uncommenting the following:

```
#install specific version of libraries used in  lab
#! mamba install pandas==1.3.3  -y
#! mamba install numpy=1.21.2 -y

# import pandas library
import pandas as pd
import numpy as np
```

This function will download the dataset into your browser

```
#This function will download the dataset into your browser

from pyodide.http import pyfetch

async def download(url, filename):
    response = await pyfetch(url)
    if response.status == 200:
        with open(filename, "wb") as f:
            f.write(await response.bytes())
```

```
path = "https://cf-courses-data.s3.us.cloud-object-
storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-
SkillsNetwork/labs/Data%20files/auto.csv"
```

you will need to download the dataset; if you are running locally, please comment out the following

```
#you will need to download the dataset; if you are running locally,
please comment out the following
await download(path, "auto.csv")
path="auto.csv"
```

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

```
# Import pandas library
import pandas as pd

# Read the online file by the URL provides above, and assign it to
variable "df"

df = pd.read_csv(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.

```
# 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

   0    1            2    3    4     5            6    7      8     9
...  \
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   2  164         audi  gas  std  four        sedan  4wd  front  99.4
...

    16    17    18    19    20   21    22  23  24     25
0  130  mpfi  3.47  2.68   9.0  111  5000  21  27  13495
1  130  mpfi  3.47  2.68   9.0  111  5000  21  27  16500
2  152  mpfi  2.68  3.47   9.0  154  5000  19  26  16500
```

```
3  109  mpfi  3.19  3.40  10.0  102  5500  24  30  13950
4  136  mpfi  3.19  3.40   8.0  115  5500  18  22  17450

[5 rows x 26 columns]
```

```
# Write your code below and press Shift+Enter to execute
print("The last 10 rows of the dataframe\n")
df.tail(10)
```

```
The last 10 rows of the dataframe


     0    1      2       3      4     5      6    7      8
9   ...   16  \
195  -1   74  volvo     gas    std  four  wagon  rwd  front
104.3  ...  141
196  -2  103  volvo     gas    std  four  sedan  rwd  front
104.3  ...  141
197  -1   74  volvo     gas    std  four  wagon  rwd  front
104.3  ...  141
198  -2  103  volvo     gas  turbo  four  sedan  rwd  front
104.3  ...  130
199  -1   74  volvo     gas  turbo  four  wagon  rwd  front
104.3  ...  130
200  -1   95  volvo     gas    std  four  sedan  rwd  front
109.1  ...  141
201  -1   95  volvo     gas  turbo  four  sedan  rwd  front
109.1  ...  141
202  -1   95  volvo     gas    std  four  sedan  rwd  front
109.1  ...  173
203  -1   95  volvo  diesel  turbo  four  sedan  rwd  front
109.1  ...  145
204  -1   95  volvo     gas  turbo  four  sedan  rwd  front
109.1  ...  141

       17    18    19    20   21    22  23  24     25
195  mpfi  3.78  3.15   9.5  114  5400  23  28  13415
196  mpfi  3.78  3.15   9.5  114  5400  24  28  15985
197  mpfi  3.78  3.15   9.5  114  5400  24  28  16515
198  mpfi  3.62  3.15   7.5  162  5100  17  22  18420
199  mpfi  3.62  3.15   7.5  162  5100  17  22  18950
200  mpfi  3.78  3.15   9.5  114  5400  23  28  16845
201  mpfi  3.78  3.15   8.7  160  5300  19  25  19045
202  mpfi  3.58  2.87   8.8  134  5500  18  23  21485
203   idi  3.01  3.40  23.0  106  4800  26  27  22470
204  mpfi  3.78  3.15   9.5  114  5400  19  25  22625

[10 rows x 26 columns]
```

```
# 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-
type', '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 dataframe:

```
df.columns = headers
df.head(10)
```

```
   symboling normalized-losses         make fuel-type aspiration num-
of-doors  \
0          3                  ?  alfa-romero       gas        std
two
1          3                  ?  alfa-romero       gas        std
two
2          1                  ?  alfa-romero       gas        std
two
3          2                164         audi       gas        std
four
4          2                164         audi       gas        std
four
5          2                  ?         audi       gas        std
two
6          1                158         audi       gas        std
four
7          1                  ?         audi       gas        std
four
8          1                158         audi       gas      turbo
four
9          0                  ?         audi       gas      turbo
two

    body-style drive-wheels engine-location  wheel-base  ...  engine-
size  \
0  convertible          rwd           front        88.6  ...
130
1  convertible          rwd           front        88.6  ...
```

```
130
2    hatchback          rwd           front        94.5  ...
152
3        sedan          fwd           front        99.8  ...
109
4        sedan          4wd           front        99.4  ...
136
5        sedan          fwd           front        99.8  ...
136
6        sedan          fwd           front       105.8  ...
136
7        wagon          fwd           front       105.8  ...
136
8        sedan          fwd           front       105.8  ...
131
9    hatchback          4wd           front        99.5  ...
131

   fuel-system  bore  stroke compression-ratio horsepower  peak-rpm
city-mpg  \
0         mpfi  3.47    2.68               9.0        111      5000
21
1         mpfi  3.47    2.68               9.0        111      5000
21
2         mpfi  2.68    3.47               9.0        154      5000
19
3         mpfi  3.19    3.40              10.0        102      5500
24
4         mpfi  3.19    3.40               8.0        115      5500
18
5         mpfi  3.19    3.40               8.5        110      5500
19
6         mpfi  3.19    3.40               8.5        110      5500
19
7         mpfi  3.19    3.40               8.5        110      5500
19
8         mpfi  3.13    3.40               8.3        140      5500
17
9         mpfi  3.13    3.40               7.0        160      5500
16

  highway-mpg  price
0          27  13495
1          27  16500
2          26  16500
3          30  13950
4          22  17450
5          25  15250
6          25  17710
```

```
7          25  18920
8          20  23875
9          22      ?

[10 rows x 26 columns]
```

We need to replace the "?" symbol with NaN so the dropna() can remove the missing values:

```
df1=df.replace('?',np.NaN)
```

We can drop missing values along the column "price" as follows:

```
df=df1.dropna(subset=["price"], axis=0)
df.head(20)
```

```
    symboling normalized-losses         make fuel-type aspiration  \
0           3                NaN  alfa-romero       gas        std
1           3                NaN  alfa-romero       gas        std
2           1                NaN  alfa-romero       gas        std
3           2                164         audi       gas        std
4           2                164         audi       gas        std
5           2                NaN         audi       gas        std
6           1                158         audi       gas        std
7           1                NaN         audi       gas        std
8           1                158         audi       gas      turbo
10          2                192          bmw       gas        std
11          0                192          bmw       gas        std
12          0                188          bmw       gas        std
13          0                188          bmw       gas        std
14          1                NaN          bmw       gas        std
15          0                NaN          bmw       gas        std
16          0                NaN          bmw       gas        std
17          0                NaN          bmw       gas        std
18          2                121    chevrolet       gas        std
19          1                 98    chevrolet       gas        std
20          0                 81    chevrolet       gas        std

   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
...
```

```
5           two         sedan           fwd            front         99.8
...
6          four         sedan           fwd            front        105.8
...
7          four         wagon           fwd            front        105.8
...
8          four         sedan           fwd            front        105.8
...
10          two         sedan           rwd            front        101.2
...
11         four         sedan           rwd            front        101.2
...
12          two         sedan           rwd            front        101.2
...
13         four         sedan           rwd            front        101.2
...
14         four         sedan           rwd            front        103.5
...
15         four         sedan           rwd            front        103.5
...
16          two         sedan           rwd            front        103.5
...
17         four         sedan           rwd            front        110.0
...
18          two     hatchback           fwd            front         88.4
...
19          two     hatchback           fwd            front         94.5
...
20         four         sedan           fwd            front         94.5
...

    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
5           136         mpfi  3.19    3.40               8.5
110
6           136         mpfi  3.19    3.40               8.5
110
7           136         mpfi  3.19    3.40               8.5
110
```

```
8            131          mpfi  3.13    3.40                 8.3
140
10           108          mpfi  3.50    2.80                 8.8
101
11           108          mpfi  3.50    2.80                 8.8
101
12           164          mpfi  3.31    3.19                 9.0
121
13           164          mpfi  3.31    3.19                 9.0
121
14           164          mpfi  3.31    3.19                 9.0
121
15           209          mpfi  3.62    3.39                 8.0
182
16           209          mpfi  3.62    3.39                 8.0
182
17           209          mpfi  3.62    3.39                 8.0
182
18            61          2bbl  2.91    3.03                 9.5
48
19            90          2bbl  3.03    3.11                 9.6
70
20            90          2bbl  3.03    3.11                 9.6
70

    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
5       5500       19           25  15250
6       5500       19           25  17710
7       5500       19           25  18920
8       5500       17           20  23875
10      5800       23           29  16430
11      5800       23           29  16925
12      4250       21           28  20970
13      4250       21           28  21105
14      4250       20           25  24565
15      5400       16           22  30760
16      5400       16           22  41315
17      5400       15           20  36880
18      5100       47           53   5151
19      5400       38           43   6295
20      5400       38           43   6575

[20 rows x 26 columns]
```

Now, we have successfully read the raw dataset and added the correct headers into the dataframe.

Question #2: Find the name of the columns of the dataframe.

```
# Write your code below and press Shift+Enter to execute
print(df.columns)
```

```
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')
```

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

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()`
...	...	...

```
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
```

A series with the data type of each column is returned.

```
# 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
```

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

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

```
df.describe()
```

```
        symboling  wheel-base      length       width      height  \
count  201.000000  201.000000  201.000000  201.000000  201.000000
mean     0.840796   98.797015  174.200995   65.889055   53.766667
std      1.254802    6.066366   12.322175    2.101471    2.447822
min     -2.000000   86.600000  141.100000   60.300000   47.800000
25%      0.000000   94.500000  166.800000   64.100000   52.000000
50%      1.000000   97.000000  173.200000   65.500000   54.100000
75%      2.000000  102.400000  183.500000   66.600000   55.500000
max      3.000000  120.900000  208.100000   72.000000   59.800000

       curb-weight  engine-size  compression-ratio    city-mpg
highway-mpg
count   201.000000   201.000000         201.000000  201.000000
201.000000
mean   2555.666667   126.875622          10.164279   25.179104
30.686567
std     517.296727    41.546834           4.004965    6.423220
6.815150
min    1488.000000    61.000000           7.000000   13.000000
16.000000
25%    2169.000000    98.000000           8.600000   19.000000
25.000000
50%    2414.000000   120.000000           9.000000   24.000000
30.000000
75%    2926.000000   141.000000           9.400000   30.000000
34.000000
max    4066.000000   326.000000          23.000000   49.000000
54.000000
```

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

```
        symboling normalized-losses    make fuel-type aspiration  \
count   201.000000               164     201       201        201
unique         NaN                51      22         2          2
top            NaN               161  toyota       gas        std
freq           NaN                11      32       181        165
mean      0.840796               NaN     NaN       NaN        NaN
std       1.254802               NaN     NaN       NaN        NaN
min      -2.000000               NaN     NaN       NaN        NaN
25%       0.000000               NaN     NaN       NaN        NaN
50%       1.000000               NaN     NaN       NaN        NaN
75%       2.000000               NaN     NaN       NaN        NaN
max       3.000000               NaN     NaN       NaN        NaN

       num-of-doors body-style drive-wheels engine-location  wheel-
base  ...  \
count           199        201          201             201
```

```
201.000000  ...
unique            2          5            3               2
NaN  ...
top            four      sedan          fwd           front
NaN  ...
freq            113         94          118             198
NaN  ...
mean            NaN        NaN          NaN             NaN
98.797015  ...
std             NaN        NaN          NaN             NaN
6.066366  ...
min             NaN        NaN          NaN             NaN
86.600000  ...
25%             NaN        NaN          NaN             NaN
94.500000  ...
50%             NaN        NaN          NaN             NaN
97.000000  ...
75%             NaN        NaN          NaN             NaN
102.400000  ...
max             NaN        NaN          NaN             NaN
120.900000  ...

        engine-size  fuel-system  bore  stroke compression-ratio
horsepower  \
count    201.000000          201   197     197        201.000000
199
unique          NaN            8    38      36               NaN
58
top             NaN         mpfi  3.62    3.40               NaN
68
freq            NaN           92    23      19               NaN
19
mean     126.875622          NaN   NaN     NaN         10.164279
NaN
std       41.546834          NaN   NaN     NaN          4.004965
NaN
min       61.000000          NaN   NaN     NaN          7.000000
NaN
25%       98.000000          NaN   NaN     NaN          8.600000
NaN
50%      120.000000          NaN   NaN     NaN          9.000000
NaN
75%      141.000000          NaN   NaN     NaN          9.400000
NaN
max      326.000000          NaN   NaN     NaN         23.000000
NaN

        peak-rpm    city-mpg highway-mpg price
count        199  201.000000  201.000000   201
```

```
unique        22          NaN          NaN   186
top         5500          NaN          NaN  8921
freq          36          NaN          NaN     2
mean         NaN   25.179104   30.686567   NaN
std          NaN    6.423220    6.815150   NaN
min          NaN   13.000000   16.000000   NaN
25%          NaN   19.000000   25.000000   NaN
50%          NaN   24.000000   30.000000   NaN
75%          NaN   30.000000   34.000000   NaN
max          NaN   49.000000   54.000000   NaN

[11 rows x 26 columns]
```

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

```
           length  compression-ratio
count  201.000000         201.000000
mean   174.200995          10.164279
std     12.322175           4.004965
min    141.100000           7.000000
25%    166.800000           8.600000
50%    173.200000           9.000000
75%    183.500000           9.400000
max    208.100000          23.000000
```

Another method you can use to check your dataset is:

It provides a concise summary of your DataFrame.

This method prints information about a DataFrame including the index dtype and columns, non-null values and memory usage.

```
# look at the info of "df"
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 201 entries, 0 to 204
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   symboling          201 non-null    int64
 1   normalized-losses  164 non-null    object
 2   make               201 non-null    object
 3   fuel-type          201 non-null    object
 4   aspiration         201 non-null    object
 5   num-of-doors       199 non-null    object
 6   body-style         201 non-null    object
 7   drive-wheels       201 non-null    object
 8   engine-location    201 non-null    object
```

```
 9   wheel-base          201 non-null    float64
 10  length              201 non-null    float64
 11  width               201 non-null    float64
 12  height              201 non-null    float64
 13  curb-weight         201 non-null    int64
 14  engine-type         201 non-null    object
 15  num-of-cylinders    201 non-null    object
 16  engine-size         201 non-null    int64
 17  fuel-system         201 non-null    object
 18  bore                197 non-null    object
 19  stroke              197 non-null    object
 20  compression-ratio   201 non-null    float64
 21  horsepower          199 non-null    object
 22  peak-rpm            199 non-null    object
 23  city-mpg            201 non-null    int64
 24  highway-mpg         201 non-null    int64
 25  price               201 non-null    object
dtypes: float64(5), int64(5), object(16)
memory usage: 29.8+ KB
```

Thank you for completing this lab!

Author

Joseph Santarcangelo

Other Contributors

Mahdi Noorian PhD

Bahare Talayian

Eric Xiao

Steven Dong

Parizad

Hima Vasudevan

Fiorella Wenver

Yi Yao.

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2022-08-23	2.4	Malika	Import micropip added and parameter for ipwidgets and tqdm

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2020-10-30	2.3	Lakshmi	Changed URL of the csv
2020-09-22	2.2	Nayef	Added replace() method to remove '?'
2020-09-09	2.1	Lakshmi	Made changes in info method of dataframe
2020-08-27	2.0	Lavanya	Moved lab to course repo in GitLab