Unit-1 Introduction to Python Pandas

In [1]:
```
pip install pandas
```

```
Requirement already satisfied: pandas in c:\programdata\anaconda3\lib\site-packages (1.1.3)
Requirement already satisfied: pytz>=2017.2 in c:\programdata\anaconda3\lib\site-packages (from pandas) (2020.1)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\programdata\anaconda3\lib\site-packages (from pandas) (2.8.
1)
Requirement already satisfied: numpy>=1.15.4 in c:\programdata\anaconda3\lib\site-packages (from pandas) (1.19.2)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-packages (from python-dateutil>=2.7.3->pan
das) (1.15.0)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
import pandas as pd
print(pd.__version__) # 1.1.3
```

```
1.1.3
```

Series :

- List : support
- Tuple : support
- Set : not support
- Dictnory : support
- if ? : that given object

In [3]:
```
# Series :- 1 column,many rows & 1D
import pandas as pd
a = [1,2,3]
myvar = pd.Series(a)
print(myvar)
print(myvar[0])
print(myvar[1])
print(myvar[2])

# Output :

# 0    1
# 1    2
# 2    3
# dtype: int64
```

```
0    1
1    2
2    3
dtype: int64
```

In [10]:
```
# Task For Tuple
import pandas as pd
a = (1,2,3)
myvar = pd.Series(a)
print(myvar)
print(myvar[0])
print(myvar[1])
print(myvar[2])
# print(myvar[3]) # Key Error
```

```
0    1
1    2
2    3
dtype: int64
1
2
3
```

In [6]:
```python
# For another data type
import pandas as pd
a = (1.0,2.0,3.0)
myvar = pd.Series(a)
print(myvar)
print(myvar[0])
print(myvar[1])
print(myvar[2])
```

```
0    1.0
1    2.0
2    3.0
dtype: float64
1.0
2.0
3.0
```

In [7]:
```python
import pandas as pd
a = (1.0,2,3.0) # convert to float
myvar = pd.Series(a)
print(myvar)
print(myvar[0])
print(myvar[1])
print(myvar[2])
```

```
0    1.0
1    2.0
2    3.0
dtype: float64
1.0
2.0
3.0
```

In [8]:
```python
import pandas as pd
a = (1.0,2,'a') # give object datatype if any character input
myvar = pd.Series(a)
print(myvar)
print(myvar[0])
print(myvar[1])
print(myvar[2])
```

```
0    1
1    2
2    a
dtype: object
1.0
2
a
```

In [9]:
```python
import pandas as pd
a = {1.0,2,'a'}
myvar = pd.Series(a)
print(myvar)
print(myvar[0])
print(myvar[1])
print(myvar[2]) # TypeError: 'set' type is unordered
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-9-b86b612605b6> in <module>
      1 import pandas as pd
      2 a = {1.0,2,'a'}
----> 3 myvar = pd.Series(a)
      4 print(myvar)
      5 print(myvar[0])

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\series.py in __init__(self, data, index, dtype, name, copy, fast
path)
    297                 pass
    298             elif isinstance(data, (set, frozenset)):
--> 299                 raise TypeError(f"'{type(data).__name__}' type is unordered")
    300             else:
    301                 data = com.maybe_iterable_to_list(data)

TypeError: 'set' type is unordered
```

In [13]:
```
import pandas as pd
a = {'A':1,'B':2,'C':3}
myvar = pd.Series(a)
print(myvar)
print(myvar['A'])
print(myvar['B'])
print(myvar['C'])
```

```
A    1
B    2
C    3
dtype: int64
1
2
3
```

In [14]:
```
import pandas as pd
a = {'A':[1,2],'B':2,'C':3} # if we pass dictniory in list give object.
myvar = pd.Series(a)
print(myvar)
print(myvar['A'])
print(myvar['B'])
print(myvar['C'])
```

```
A    [1, 2]
B         2
C         3
dtype: object
[1, 2]
2
3
```

In [16]:
```
import pandas as pd
a = [1,2,3]
myvar = pd.Series(a,index=['x','y']) # we have must pass 3 values.
print(myvar) # ValueError: Length of passed values is 3, index implies 2.

```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-16-349654d3951c> in <module>
      1 import pandas as pd
      2 a = [1,2,3]
----> 3 myvar = pd.Series(a,index=['x','y']) # we have must pass 3 values.
      4 print(myvar) # ValueError: Length of passed values is 3, index implies 2.

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\series.py in __init__(self, data, index, dtype, name, copy, fast
path)
    311                 try:
    312                     if len(index) != len(data):
--> 313                         raise ValueError(
    314                             f"Length of passed values is {len(data)}, "
    315                             f"index implies {len(index)}."

ValueError: Length of passed values is 3, index implies 2.
```

In [17]:
```
import pandas as pd
a = [1,2,3]
myvar = pd.Series(a,index=['x','y','z'])
print(myvar)

# Output :
# x    1
# y    2
# z    3
# dtype: int64
```

```
x    1
y    2
z    3
dtype: int64
```

In [20]:
```
import pandas as pd
calories = {'day1':420,'day2':380,'day3':390}
myvar = pd.Series(calories)
print(myvar)

# Output :
# day1    420
# day2    380
# day3    390
# dtype: int64
```

```
day1    420
day2    380
day3    390
dtype: int64
```

In [23]:
```
import pandas as pd
calories = {'day1':420,'day2':380,'day3':390}
myvar = pd.Series(calories,index=['x','y','z'])
print(myvar)

# Output :
# x   NaN
# y   NaN
# z   NaN
# dtype: float64
```

```
x   NaN
y   NaN
z   NaN
dtype: float64
```

In [24]:
```
import pandas as pd
calories = {'day1':420,'day2':380,'day3':390}
myvar = pd.Series(calories,index=['x','y','z','day1'])
print(myvar) # NaN convert automacally float.

# Output :
# x         NaN
# y         NaN
# z         NaN
# day1    420.0
# dtype: float64
```

```
x         NaN
y         NaN
z         NaN
day1    420.0
dtype: float64
```

In [25]:
```
import pandas as pd
calories = {'day1':420,'day2':380,'day3':390}
myvar = pd.Series(calories,index=['day2','day1'])
print(myvar) # value must be same of particular key.
```

```
day2    380
day1    420
dtype: int64
```

In [30]:
```
a = [1,2,3,4,5,6]
myvar = pd.Series(a)
myvar[[0,1,3]] # when we pass multiple value then using list.
# myvar[0,1,3]
myvar[0::2]
```

Out[30]:
```
0    1
2    3
4    5
dtype: int64
```

DataFrame :(2D)

- many rows many columns

In [31]:
```python
import pandas as pd
data = {'calories':[420,380,390],'duration':[50,40,45]}
df = pd.DataFrame(data)
print(df)

#    calories  duration
# 0       420        50
# 1       380        40
# 2       390        45
```

```
   calories  duration
0       420        50
1       380        40
2       390        45
```

- loc & iloc(integer location)

- loc : accepts labels as well as int
- iloc: accepts only integer not a string

In [33]:
```python
import pandas as pd
data = {'calories':[420,380,390],'duration':[50,40,45]}
df = pd.DataFrame(data)
print(df['calories'][0])
print(df['calories'].loc[0])
print(df['duration'].loc[1])
```

```
420
420
40
```

In [3]:
```python
import pandas as pd
data = {'calories':[420,380,390],'duration':[50,40,45]}
df = pd.DataFrame(data)
print(df)
```

```
   calories  duration
0       420        50
1       380        40
2       390        45
```

In [9]:
```python
import pandas as pd
data = {'calories':[420,380,390],'duration':[50,40,45]}
df = pd.DataFrame(data,index=['day1','day2','day3'])
# print(df['calories'].loc[0]) # give key error because index is change.
print(df['calories'].loc['day1']) # 420

# Output :
#        calories  duration
# day1       420        50
# day2       380        40
# day3       390        45
```

```
420
```

In [10]:
```python
import pandas as pd
data = {'calories':[420,380,390],'duration':[50,40,45]}
df = pd.DataFrame(data)
print(df['calories'].iloc[0]) # 420
```

```
420
```

In [12]:
```python
import pandas as pd
data = {'calories':[420,380,390],'duration':[50,40,45]}
df = pd.DataFrame(data,index=['day1','day2','day3'])
# print(df['calories'].iloc['day1']) # TypeError:Cannot index by location index with a non-integer key
```

For CSV File.

In [14]:
```python
import pandas as pd
df = pd.read_csv('auto-mpg.csv') # converting into dataframe
df
```

Out[14]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino
...	...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86	2790	15.6	82	1	ford mustang gl
394	44.0	4	97.0	52	2130	24.6	82	2	vw pickup
395	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
396	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
397	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

398 rows × 9 columns

In [15]:
```python
import pandas as pd
df = pd.read_csv('auto-mpg.csv') # converting into dataframe
df.info()

# Output :

# <class 'pandas.core.frame.DataFrame'>
# RangeIndex: 398 entries, 0 to 397
# Data columns (total 9 columns):
#  #   Column        Non-Null Count  Dtype
# ---  ------        --------------  -----
#  0   mpg           398 non-null    float64
#  1   cylinders     398 non-null    int64
#  2   displacement  398 non-null    float64
#  3   horsepower    398 non-null    object
#  4   weight        398 non-null    int64
#  5   acceleration  398 non-null    float64
#  6   model year    398 non-null    int64
#  7   origin        398 non-null    int64
#  8   car name      398 non-null    object
# dtypes: float64(3), int64(4), object(2)
# memory usage: 28.1+ KB
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64
 2   displacement  398 non-null    float64
 3   horsepower    398 non-null    object
 4   weight        398 non-null    int64
 5   acceleration  398 non-null    float64
 6   model year    398 non-null    int64
 7   origin        398 non-null    int64
 8   car name      398 non-null    object
dtypes: float64(3), int64(4), object(2)
memory usage: 28.1+ KB
```

In [17]:
```
help(pd)
```

```
Help on package pandas:

NAME
    pandas

DESCRIPTION
    pandas - a powerful data analysis and manipulation library for Python
    =====================================================================

    **pandas** is a Python package providing fast, flexible, and expressive data
    structures designed to make working with "relational" or "labeled" data both
    easy and intuitive. It aims to be the fundamental high-level building block for
    doing practical, **real world** data analysis in Python. Additionally, it has
    the broader goal of becoming **the most powerful and flexible open source data
    analysis / manipulation tool available in any language**. It is already well on
    its way toward this goal.

    Main Features
    -------------
    Here are just a few of the things that pandas does well:

      - Easy handling of missing data in floating point as well as non-floating
        point data.
      - Size mutability: columns can be inserted and deleted from DataFrame and
        higher dimensional objects
      - Automatic and explicit data alignment: objects can be explicitly aligned
        to a set of labels, or the user can simply ignore the labels and let
        `Series`, `DataFrame`, etc. automatically align the data for you in
        computations.
      - Powerful, flexible group by functionality to perform split-apply-combine
        operations on data sets, for both aggregating and transforming data.
      - Make it easy to convert ragged, differently-indexed data in other Python
        and NumPy data structures into DataFrame objects.
      - Intelligent label-based slicing, fancy indexing, and subsetting of large
        data sets.
      - Intuitive merging and joining data sets.
      - Flexible reshaping and pivoting of data sets.
      - Hierarchical labeling of axes (possible to have multiple labels per tick).
      - Robust IO tools for loading data from flat files (CSV and delimited),
        Excel files, databases, and saving/loading data from the ultrafast HDF5
        format.
      - Time series-specific functionality: date range generation and frequency
        conversion, moving window statistics, date shifting and lagging.

PACKAGE CONTENTS
    _config (package)
    _libs (package)
    _testing
    _typing
    _version
    api (package)
    arrays (package)
    compat (package)
    conftest
    core (package)
    errors (package)
    io (package)
    plotting (package)
    testing
    tests (package)
    tseries (package)
    util (package)

SUBMODULES
    _hashtable
    _lib
    _tslib
    offsets

FUNCTIONS
    __getattr__(name)

DATA
    IndexSlice = <pandas.core.indexing._IndexSlice object>
    NA = <NA>
    NaT = NaT
    __docformat__ = 'restructuredtext'
    __git_version__ = 'db08276bc116c438d3fdee492026f8223584c477'
    describe_option = <pandas._config.config.CallableDynamicDoc object>
    get_option = <pandas._config.config.CallableDynamicDoc object>
    options = <pandas._config.config.DictWrapper object>
    reset_option = <pandas._config.config.CallableDynamicDoc object>
    set_option = <pandas._config.config.CallableDynamicDoc object>

VERSION
    1.1.3

FILE
```

```
c:\programdata\anaconda3\lib\site-packages\pandas\__init__.py
```

- head() & tail()

In [18]:
```python
import pandas as pd
df = pd.read_csv('auto-mpg.csv') # converting into dataframe
df.head() # given first 5 row print by default when args not pass.
```

Out[18]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino

In [19]:
```python
import pandas as pd
df = pd.read_csv('auto-mpg.csv') # converting into dataframe
df.tail() # given last 5 row print by default when args not pass.
```

Out[19]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
393	27.0	4	140.0	86	2790	15.6	82	1	ford mustang gl
394	44.0	4	97.0	52	2130	24.6	82	2	vw pickup
395	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
396	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
397	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

In [20]:
```python
import pandas as pd
df = pd.read_csv('auto-mpg.csv') # converting into dataframe
df.head(10)
```

Out[20]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino
5	15.0	8	429.0	198	4341	10.0	70	1	ford galaxie 500
6	14.0	8	454.0	220	4354	9.0	70	1	chevrolet impala
7	14.0	8	440.0	215	4312	8.5	70	1	plymouth fury iii
8	14.0	8	455.0	225	4425	10.0	70	1	pontiac catalina
9	15.0	8	390.0	190	3850	8.5	70	1	amc ambassador dpl

In [21]:
```python
import pandas as pd
df = pd.read_csv('auto-mpg.csv') # converting into dataframe
df.tail(10)
```

Out[21]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
388	26.0	4	156.0	92	2585	14.5	82	1	chrysler lebaron medallion
389	22.0	6	232.0	112	2835	14.7	82	1	ford granada l
390	32.0	4	144.0	96	2665	13.9	82	3	toyota celica gt
391	36.0	4	135.0	84	2370	13.0	82	1	dodge charger 2.2
392	27.0	4	151.0	90	2950	17.3	82	1	chevrolet camaro
393	27.0	4	140.0	86	2790	15.6	82	1	ford mustang gl
394	44.0	4	97.0	52	2130	24.6	82	2	vw pickup
395	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
396	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
397	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

In [22]:
```python
import pandas as pd
df = pd.read_csv('auto-mpg.csv') # converting into dataframe
df.loc[34:56]
```

Out[22]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
34	16.0	6	225.0	105	3439	15.5	71	1	plymouth satellite custom
35	17.0	6	250.0	100	3329	15.5	71	1	chevrolet chevelle malibu
36	19.0	6	250.0	88	3302	15.5	71	1	ford torino 500
37	18.0	6	232.0	100	3288	15.5	71	1	amc matador
38	14.0	8	350.0	165	4209	12.0	71	1	chevrolet impala
39	14.0	8	400.0	175	4464	11.5	71	1	pontiac catalina brougham
40	14.0	8	351.0	153	4154	13.5	71	1	ford galaxie 500
41	14.0	8	318.0	150	4096	13.0	71	1	plymouth fury iii
42	12.0	8	383.0	180	4955	11.5	71	1	dodge monaco (sw)
43	13.0	8	400.0	170	4746	12.0	71	1	ford country squire (sw)
44	13.0	8	400.0	175	5140	12.0	71	1	pontiac safari (sw)
45	18.0	6	258.0	110	2962	13.5	71	1	amc hornet sportabout (sw)
46	22.0	4	140.0	72	2408	19.0	71	1	chevrolet vega (sw)
47	19.0	6	250.0	100	3282	15.0	71	1	pontiac firebird
48	18.0	6	250.0	88	3139	14.5	71	1	ford mustang
49	23.0	4	122.0	86	2220	14.0	71	1	mercury capri 2000
50	28.0	4	116.0	90	2123	14.0	71	2	opel 1900
51	30.0	4	79.0	70	2074	19.5	71	2	peugeot 304
52	30.0	4	88.0	76	2065	14.5	71	2	fiat 124b
53	31.0	4	71.0	65	1773	19.0	71	3	toyota corolla 1200
54	35.0	4	72.0	69	1613	18.0	71	3	datsun 1200
55	27.0	4	97.0	60	1834	19.0	71	2	volkswagen model 111
56	26.0	4	91.0	70	1955	20.5	71	1	plymouth cricket

In [24]:
```python
import pandas as pd
df = pd.read_csv('auto-mpg.csv') # converting into dataframe
df.loc[[34,56]] # for particular row show then we pass list
```

Out[24]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
34	16.0	6	225.0	105	3439	15.5	71	1	plymouth satellite custom
56	26.0	4	91.0	70	1955	20.5	71	1	plymouth cricket

- df: df ni bajuma hamesha column ave.
- loc: loc ni bajuma hamesha row ave.

In [27]:
```python
import pandas as pd
df = pd.read_csv('auto-mpg.csv') # converting into dataframe
df['mpg'].loc[[34,56]]
```

Out[27]:
```
34    16.0
56    26.0
Name: mpg, dtype: float64
```

In [29]:
```python
import pandas as pd
df = pd.read_csv('auto-mpg.csv') # converting into dataframe
df[['mpg','displacement']].loc[[34,56]]
```

Out[29]:

	mpg	displacement
34	16.0	225.0
56	26.0	91.0

In [31]:
```python
import pandas as pd
df = pd.read_csv('auto-mpg.csv')
df[['mpg','cylinders']].loc[0:20]
```

Out[31]:

	mpg	cylinders
0	18.0	8
1	15.0	8
2	18.0	8
3	16.0	8
4	17.0	8
5	15.0	8
6	14.0	8
7	14.0	8
8	14.0	8
9	15.0	8
10	15.0	8
11	14.0	8
12	15.0	8
13	14.0	8
14	24.0	4
15	22.0	6
16	18.0	6
17	21.0	6
18	27.0	4
19	26.0	4
20	25.0	4

In [32]:
```python
import pandas as pd
df = pd.read_csv('auto-mpg.csv')
df.shape # (398, 9)
```

Out[32]:
```
(398, 9)
```

In [36]:
```
import pandas as pd
df = pd.read_csv('auto-mpg.csv')
df.loc[-1] # ValueError
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\range.py in get_loc(self, key, method, tolerance)
    354                 try:
--> 355                     return self._range.index(new_key)
    356                 except ValueError as err:

ValueError: -1 is not in range

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
<ipython-input-36-83f047b6aa65> in <module>
      1 import pandas as pd
      2 df = pd.read_csv('auto-mpg.csv')
----> 3 df.loc[-1]

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py in __getitem__(self, key)
    877
    878             maybe_callable = com.apply_if_callable(key, self.obj)
--> 879             return self._getitem_axis(maybe_callable, axis=axis)
    880
    881     def _is_scalar_access(self, key: Tuple):

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py in _getitem_axis(self, key, axis)
   1108         # fall thru to straight lookup
   1109         self._validate_key(key, axis)
-> 1110         return self._get_label(key, axis=axis)
   1111
   1112     def _get_slice_axis(self, slice_obj: slice, axis: int):

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py in _get_label(self, label, axis)
   1057     def _get_label(self, label, axis: int):
   1058         # GH#5667 this will fail if the label is not present in the axis.
-> 1059         return self.obj.xs(label, axis=axis)
   1060
   1061     def _handle_lowerdim_multi_index_axis0(self, tup: Tuple):

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py in xs(self, key, axis, level, drop_level)
   3489             loc, new_index = self.index.get_loc_level(key, drop_level=drop_level)
   3490         else:
-> 3491             loc = self.index.get_loc(key)
   3492
   3493             if isinstance(loc, np.ndarray):

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\range.py in get_loc(self, key, method, tolerance)
    355                     return self._range.index(new_key)
    356                 except ValueError as err:
--> 357                     raise KeyError(key) from err
    358             raise KeyError(key)
    359         return super().get_loc(key, method=method, tolerance=tolerance)

KeyError: -1
```

In [37]:
```
import pandas as pd
df = pd.read_csv('auto-mpg.csv')
df.loc[:-1] # only give column name.
```

Out[37]:

mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name

In [42]:
```
import pandas as pd
df = pd.read_csv('auto-mpg.csv')
print(df.loc[:-1])

# Empty DataFrame
# Columns: [mpg, cylinders, displacement, horsepower, weight, acceleration, model year, origin, car name]
# Index: []
```

```
Empty DataFrame
Columns: [mpg, cylinders, displacement, horsepower, weight, acceleration, model year, origin, car name]
Index: []
```

Statistics

- not analysis of object only Integer & Float.

In [44]:
```
import pandas as pd
df = pd.read_csv('auto-mpg.csv')
df.describe()
```

Out[44]:

	mpg	cylinders	displacement	weight	acceleration	model year	origin
count	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425879	2970.424623	15.568090	76.010050	1.572864
std	7.815984	1.701004	104.269838	846.841774	2.757689	3.697627	0.802055
min	9.000000	3.000000	68.000000	1613.000000	8.000000	70.000000	1.000000
25%	17.500000	4.000000	104.250000	2223.750000	13.825000	73.000000	1.000000
50%	23.000000	4.000000	148.500000	2803.500000	15.500000	76.000000	1.000000
75%	29.000000	8.000000	262.000000	3608.000000	17.175000	79.000000	2.000000
max	46.600000	8.000000	455.000000	5140.000000	24.800000	82.000000	3.000000

In [45]:
```
import pandas as pd
df = pd.read_csv('auto-mpg.csv')
df.describe(include="all")
```

Out[45]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
count	398.000000	398.000000	398.000000	398	398.000000	398.000000	398.000000	398.000000	398
unique	NaN	NaN	NaN	94	NaN	NaN	NaN	NaN	305
top	NaN	NaN	NaN	150	NaN	NaN	NaN	NaN	ford pinto
freq	NaN	NaN	NaN	22	NaN	NaN	NaN	NaN	6
mean	23.514573	5.454774	193.425879	NaN	2970.424623	15.568090	76.010050	1.572864	NaN
std	7.815984	1.701004	104.269838	NaN	846.841774	2.757689	3.697627	0.802055	NaN
min	9.000000	3.000000	68.000000	NaN	1613.000000	8.000000	70.000000	1.000000	NaN
25%	17.500000	4.000000	104.250000	NaN	2223.750000	13.825000	73.000000	1.000000	NaN
50%	23.000000	4.000000	148.500000	NaN	2803.500000	15.500000	76.000000	1.000000	NaN
75%	29.000000	8.000000	262.000000	NaN	3608.000000	17.175000	79.000000	2.000000	NaN
max	46.600000	8.000000	455.000000	NaN	5140.000000	24.800000	82.000000	3.000000	NaN

In [ ]:
```

```

In [ ]:
```

```