Unit-1 Introduction to Python Pandas

In [ ]:

```
pip install pandas
```

In [39]:

```
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 [40]:

```
# 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
1
2
3
```

In [41]:

```
# 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 [42]:

```
# 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 [43]:

```
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 [44]:

```
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 [45]:

```
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-45-5074ab57916c> 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 [46]:

```
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 [47]:

```python
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 [48]:

```python
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-48-349654d3951c> 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 [49]:

```python
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 [50]:

```python
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 [51]:

```python
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 [52]:

```python
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 [53]:

```python
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 [54]:

```python
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[54]:

```
0    1
2    3
4    5
dtype: int64
```

DataFrame :(2D)

- many rows many columns

In [55]:

```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 [56]:

```
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 [57]:

```
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 [58]:

```
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 [59]:

```
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 [60]:

```
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 [61]:

```
import pandas as pd
df = pd.read_csv('auto-mpg.csv') # converting into dataframe
df
```

Out[61]:

	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 [62]:

```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 [63]:

```
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 [64]:

```
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[64]:

	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 [65]:

```
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[65]:

	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 [66]:

```
import pandas as pd
df = pd.read_csv('auto-mpg.csv') # converting into dataframe
df.head(10)
```

Out[66]:

	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 [67]:

```
import pandas as pd
df = pd.read_csv('auto-mpg.csv') # converting into dataframe
df.tail(10)
```

Out[67]:

	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 [68]:

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

Out[68]:

	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 [69]:

```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[69]:

	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 [70]:

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

Out[70]:

```
34    16.0
56    26.0
Name: mpg, dtype: float64
```

In [71]:

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

Out[71]:

	mpg	displacement
34	16.0	225.0
56	26.0	91.0

In [72]:

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

Out[72]:

	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 [73]:

```
import pandas as pd
df = pd.read_csv('auto-mpg.csv')
df.shape # (398, 9)
```

Out[73]:

```
(398, 9)
```

In [74]:

```python
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-74-124c5c46c56e> in <module>
      1 import pandas as pd
      2 df = pd.read_csv('auto-mpg.csv')
----> 3 df.loc[-1] # ValueError

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 [75]:

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

Out[75]:

mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name

In [76]:

```python
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 [77]:

```
import pandas as pd
df = pd.read_csv('auto-mpg.csv')
df.describe()
```

Out[77]:

	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

Statistics All Operations :

In [78]:

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

Out[78]:

	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 [79]:

```
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df.describe(include=[np.number])
# df.describe(include=[np.object_]) # For Laptop[np.object]
```

Out[79]:

	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 [80]:

```
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df.describe(include=[np.object])
```

Out[80]:

	horsepower	car name
count	398	398
unique	94	305
top	150	ford pinto
freq	22	6

In [81]:

```python
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df.describe(exclude=[np.number])
```

Out[81]:

	horsepower	car name
count	398	398
unique	94	305
top	150	ford pinto
freq	22	6

In [82]:

```python
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df.describe(exclude=[np.object])
```

Out[82]:

	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 [83]:

```python
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df.describe(exclude=["O"])

#  "O" = Object.
```

Out[83]:

	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 [84]:

```python
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df["mpg"].describe()
```

Out[84]:

```
count    398.000000
mean      23.514573
std        7.815984
min        9.000000
25%       17.500000
50%       23.000000
75%       29.000000
max       46.600000
Name: mpg, dtype: float64
```

In [85]:

```
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df.loc[0:5].describe()
```

Out[85]:

	mpg	cylinders	displacement	weight	acceleration	model year	origin
count	6.000000	6.0	6.000000	6.000000	6.000000	6.0	6.0
mean	16.500000	8.0	335.000000	3642.666667	11.166667	70.0	1.0
std	1.378405	0.0	49.363954	355.980149	0.816497	0.0	0.0
min	15.000000	8.0	302.000000	3433.000000	10.000000	70.0	1.0
25%	15.250000	8.0	304.750000	3439.250000	10.625000	70.0	1.0
50%	16.500000	8.0	312.500000	3476.500000	11.250000	70.0	1.0
75%	17.750000	8.0	342.000000	3645.750000	11.875000	70.0	1.0
max	18.000000	8.0	429.000000	4341.000000	12.000000	70.0	1.0

In [86]:

```
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df.iloc[0:5].describe()
```

Out[86]:

	mpg	cylinders	displacement	weight	acceleration	model year	origin
count	5.00000	5.0	5.000000	5.000000	5.00000	5.0	5.0
mean	16.80000	8.0	316.200000	3503.000000	11.40000	70.0	1.0
std	1.30384	0.0	19.879638	110.006818	0.65192	0.0	0.0
min	15.00000	8.0	302.000000	3433.000000	10.50000	70.0	1.0
25%	16.00000	8.0	304.000000	3436.000000	11.00000	70.0	1.0
50%	17.00000	8.0	307.000000	3449.000000	11.50000	70.0	1.0
75%	18.00000	8.0	318.000000	3504.000000	12.00000	70.0	1.0
max	18.00000	8.0	350.000000	3693.000000	12.00000	70.0	1.0

In [87]:

```
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df["mpg"].loc[0:5].describe()
```

Out[87]:

```
count     6.000000
mean     16.500000
std       1.378405
min      15.000000
25%      15.250000
50%      16.500000
75%      17.750000
max      18.000000
Name: mpg, dtype: float64
```

In [88]:

```
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df.describe(percentiles=[0.3,0.57,0.83])
```

Out[88]:

	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
30%	18.000000	4.000000	112.000000	2301.000000	14.200000	73.000000	1.000000
50%	23.000000	4.000000	148.500000	2803.500000	15.500000	76.000000	1.000000
57.0%	24.358000	6.000000	187.350000	2969.060000	15.900000	77.000000	1.000000
83%	31.951000	8.000000	318.000000	3940.000000	18.151000	80.000000	3.000000
max	46.600000	8.000000	455.000000	5140.000000	24.800000	82.000000	3.000000

Corr :- corelation of cofficient

In [3]:

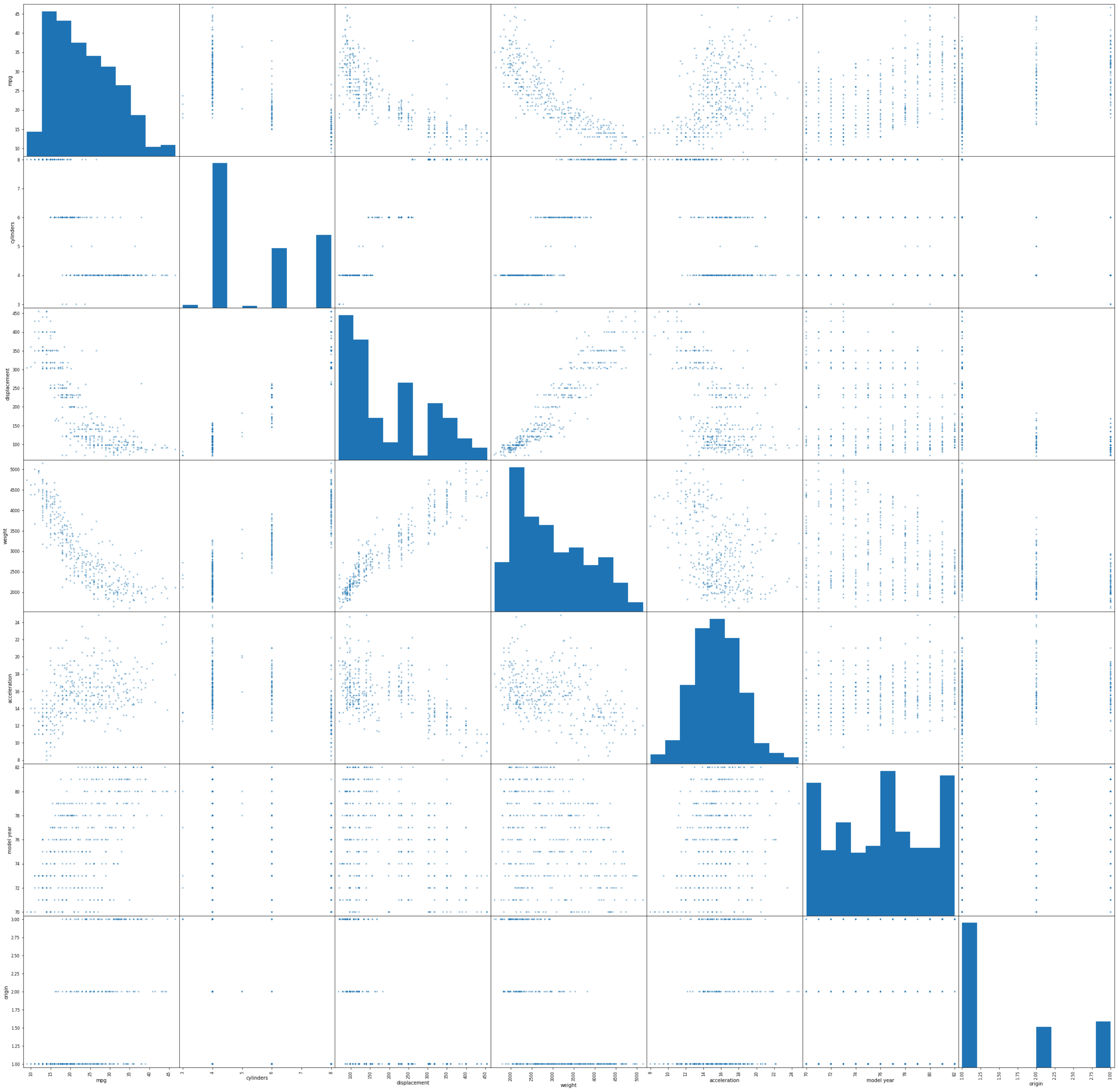
```python
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df.corr()
# df.corr(numeric_only=True)
```

Out[3]:

	mpg	cylinders	displacement	weight	acceleration	model year	origin
mpg	1.000000	-0.775396	-0.804203	-0.831741	0.420289	0.579267	0.563450
cylinders	-0.775396	1.000000	0.950721	0.896017	-0.505419	-0.348746	-0.562543
displacement	-0.804203	0.950721	1.000000	0.932824	-0.543684	-0.370164	-0.609409
weight	-0.831741	0.896017	0.932824	1.000000	-0.417457	-0.306564	-0.581024
acceleration	0.420289	-0.505419	-0.543684	-0.417457	1.000000	0.288137	0.205873
model year	0.579267	-0.348746	-0.370164	-0.306564	0.288137	1.000000	0.180662
origin	0.563450	-0.562543	-0.609409	-0.581024	0.205873	0.180662	1.000000

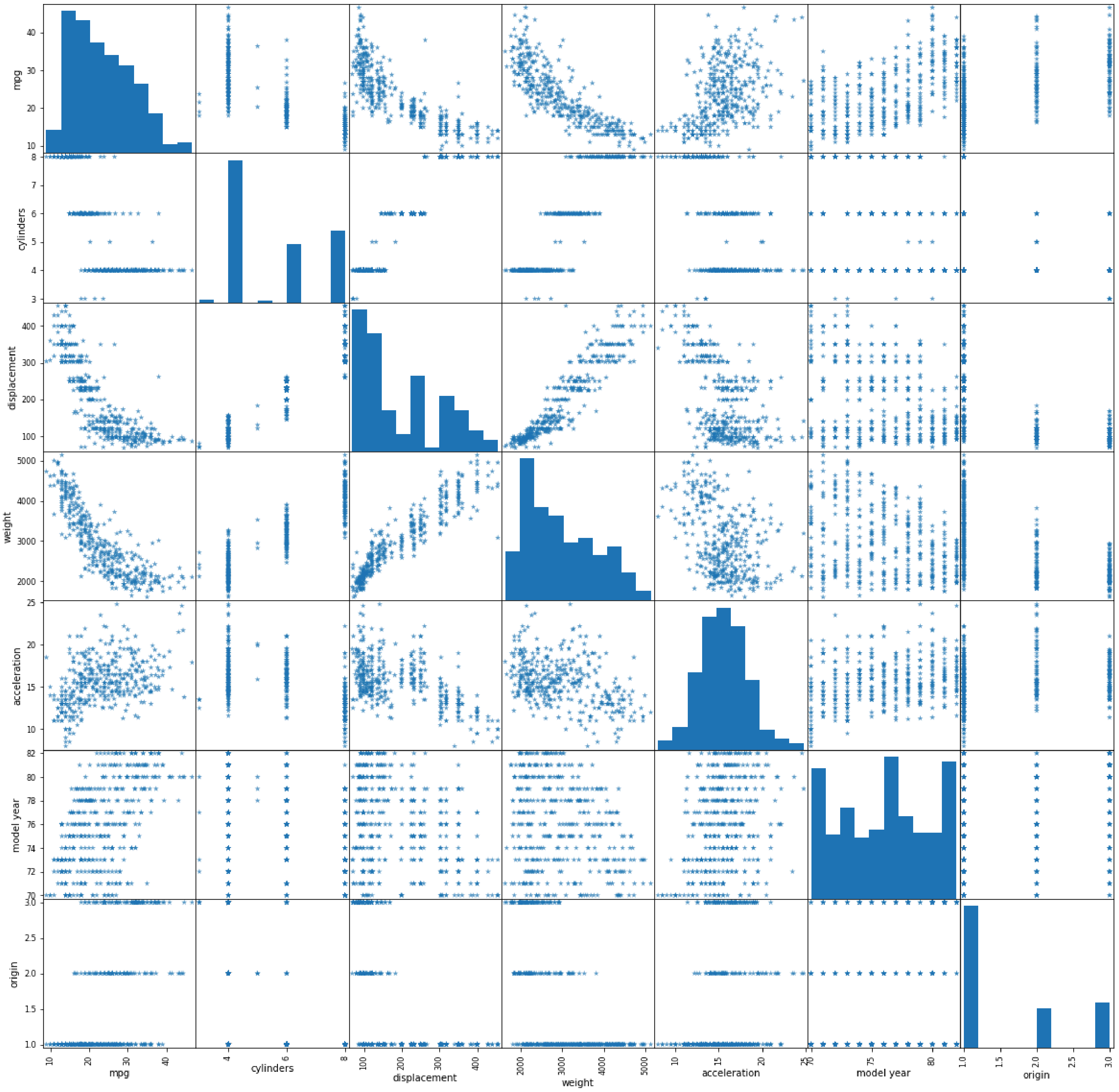
In [2]:

```python
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('auto-mpg.csv')
pd.plotting.scatter_matrix(df,figsize=[40,40])
plt.show()
```



In [4]:

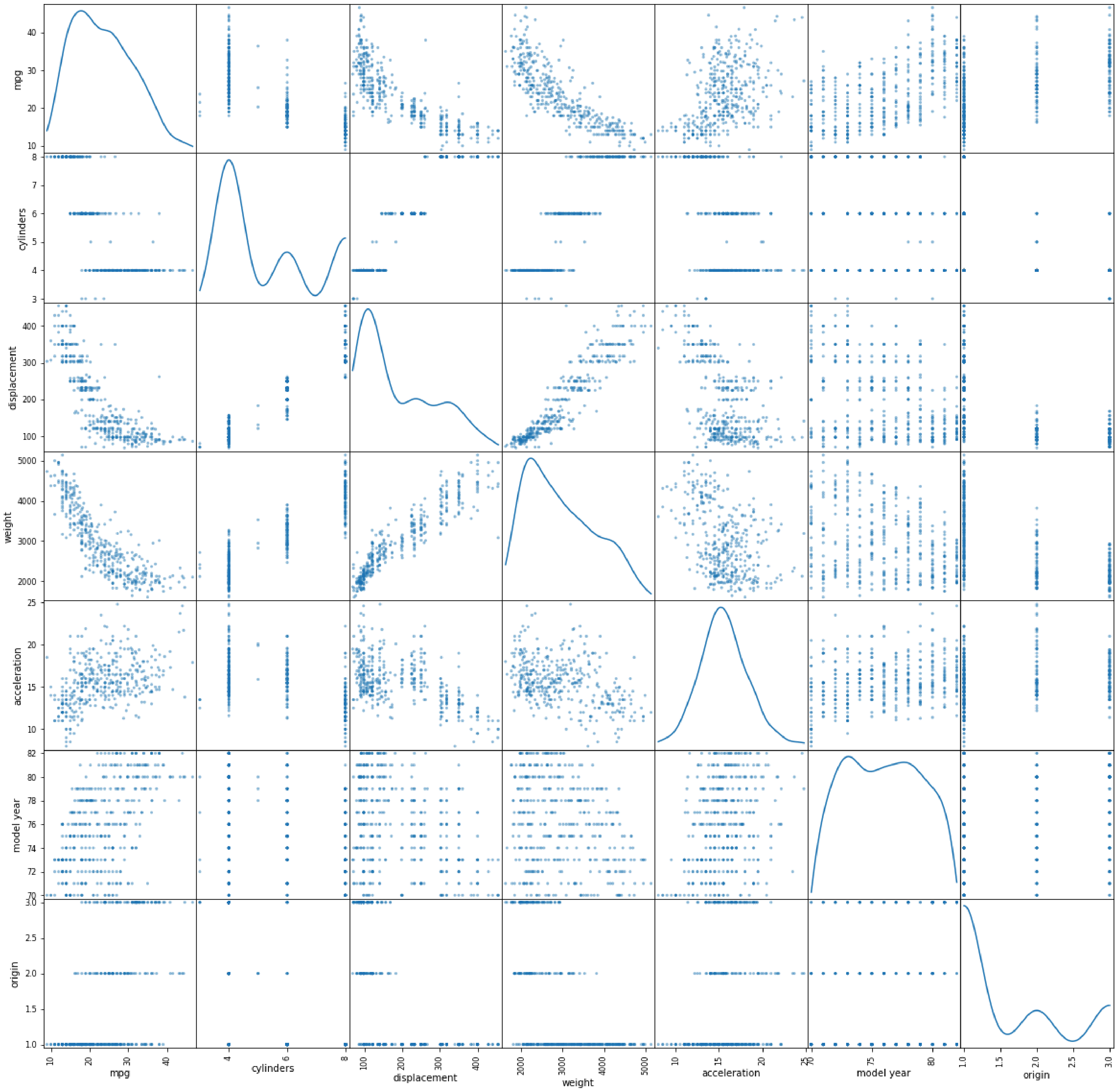
```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('auto-mpg.csv')
pd.plotting.scatter_matrix(df,figsize=[20,20],marker="*",alpha=0.7)
plt.show()
```



In [5]:

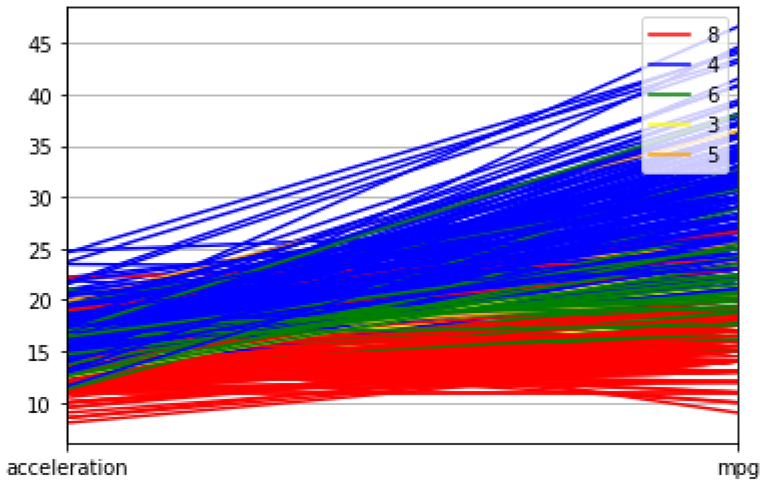
```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('auto-mpg.csv')
pd.plotting.scatter_matrix(df,figsize=[20,20],diagonal="kde")
plt.show()

# kde = kernal density estimator.
```



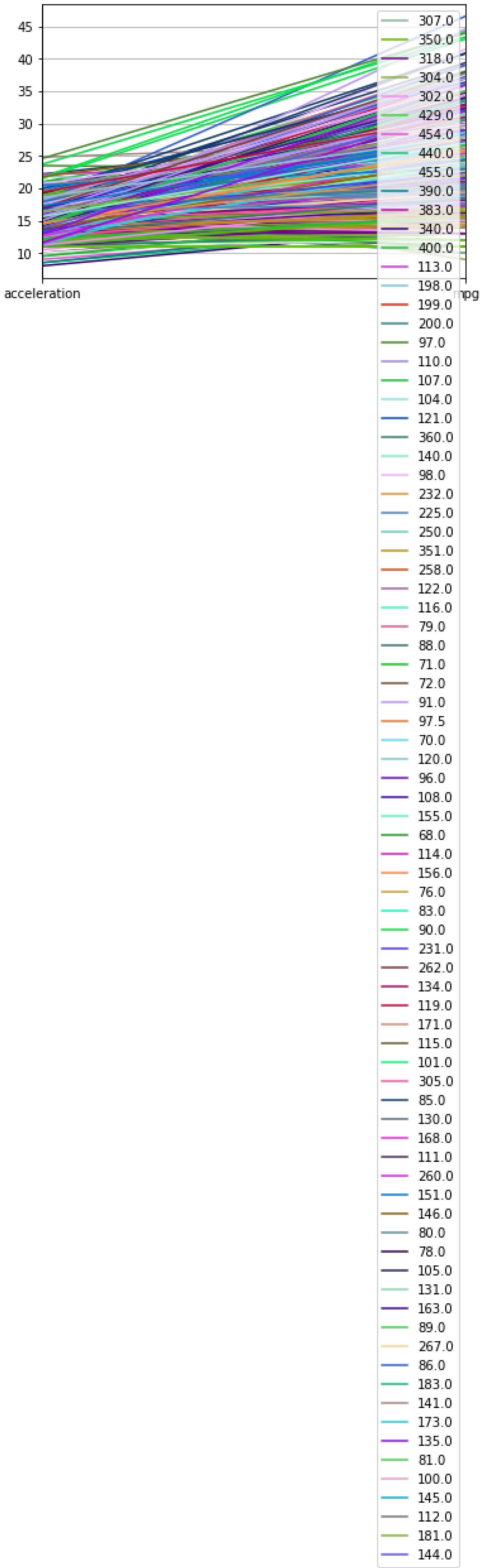
In [6]:

```
import pandas as pd
from pandas.plotting import parallel_coordinates
df = pd.read_csv('auto-mpg.csv')
pll = parallel_coordinates(df,"cylinders",cols=["acceleration","mpg"],color=["red","blue","green","yellow","orange"]
```



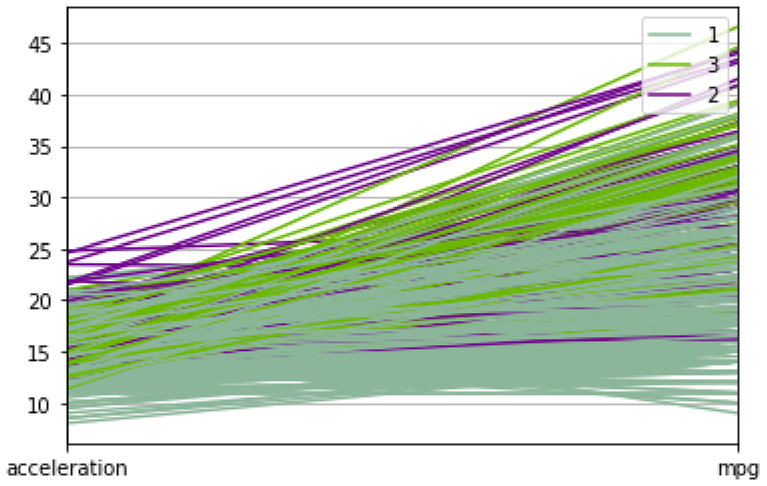
In [1]:

```python
import pandas as pd
from pandas.plotting import parallel_coordinates
df = pd.read_csv('auto-mpg.csv')
pll = parallel_coordinates(df,"displacement",cols=["acceleration","mpg"])
```



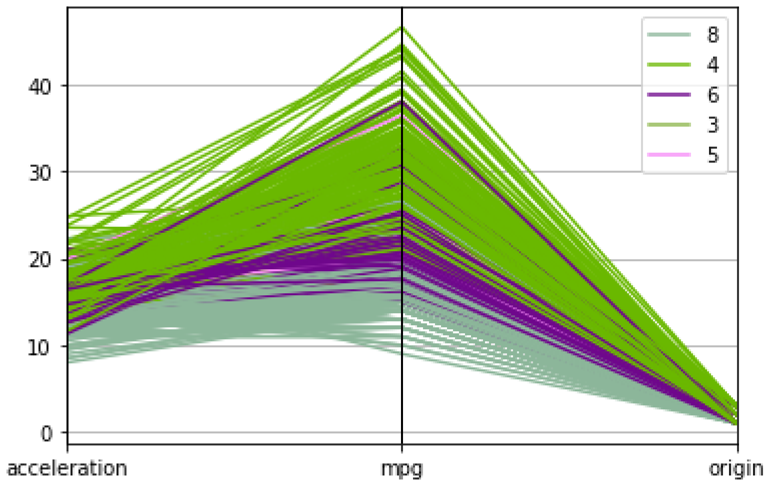
In [7]:

```python
import pandas as pd
from pandas.plotting import parallel_coordinates
df = pd.read_csv('auto-mpg.csv')
pll = parallel_coordinates(df,"origin",cols=["acceleration","mpg"])
```



In [8]:

```python
import pandas as pd
from pandas.plotting import parallel_coordinates
df = pd.read_csv('auto-mpg.csv')
pll = parallel_coordinates(df,"cylinders",cols=["acceleration","mpg","origin"])
# pll = parallel_coordinates(df,"model year",cols=["acceleration","mpg","origin"])
```



In [9]:

```python
import pandas as pd
df = pd.read_csv('auto-mpg.csv')
pd.crosstab(df["cylinders"],df["model year"],
rownames = ["cylinders"],colnames=["model year"])
```

Out[9]:

```
model year  70  71  72  73  74  75  76  77  78  79  80  81  82
 cylinders
         3   0   0   1   1   0   0   0   1   0   0   1   0   0
         4   7  13  14  11  15  12  15  14  17  12  25  21  28
         5   0   0   0   0   0   0   0   0   1   1   1   0   0
         6   4   8   0   8   7  12  10   5  12   6   2   7   3
         8  18   7  13  20   5   6   9   8   6  10   0   1   0
```

Data cleaning :-

In [10]:

```python
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,np.nan,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data
```

Out[10]:

```
   name  region  sales  expense
0     a      ma   10.0     50.0
1   NaN     NaN    NaN      NaN
2   NaN      mp   30.0     70.0
3     d      gu    NaN      NaN
4     e     NaN   50.0     90.0
```

In [11]:

```python
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                            'region':['ma',np.nan,'mp','gu',np.nan],
                            'sales':[10,np.nan,30,np.nan,50],
                            'expense':[50,np.nan,70,np.nan,90]})
sales_data.isna() # boolean type
```

Out[11]:

	name	region	sales	expense
0	False	False	False	False
1	True	True	True	True
2	True	False	False	False
3	False	False	True	True
4	False	True	False	False

In [12]:

```python
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                            'region':['ma',np.nan,'mp','gu',np.nan],
                            'sales':[10,np.nan,30,np.nan,50],
                            'expense':[50,np.nan,70,np.nan,90]})
sales_data.isna().sum()
```

Out[12]:

```
name       2
region     2
sales      2
expense    2
dtype: int64
```

In [13]:

```python
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                            'region':['ma',np.nan,'mp','gu',np.nan],
                            'sales':[10,np.nan,30,np.nan,50],
                            'expense':[50,np.nan,70,np.nan,90]})
sales_data.dropna() # nan value row remove.
```

Out[13]:

	name	region	sales	expense
0	a	ma	10.0	50.0

In [14]:

```python
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                            'region':['ma',np.nan,'mp','gu',np.nan],
                            'sales':[10,np.nan,30,np.nan,50],
                            'expense':[50,np.nan,70,np.nan,90]})
sales_data.dropna(how="all") # how=all :- je row ma badha nan hoy to ej row kadhe.
```

Out[14]:

	name	region	sales	expense
0	a	ma	10.0	50.0
2	NaN	mp	30.0	70.0
3	d	gu	NaN	NaN
4	e	NaN	50.0	90.0

In [15]:

```python
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                            'region':['ma',np.nan,'mp','gu',np.nan],
                            'sales':[10,np.nan,30,np.nan,50],
                            'expense':[50,np.nan,70,np.nan,90]})
sales_data.dropna(how="any")
```

Out[15]:

	name	region	sales	expense
0	a	ma	10.0	50.0

In [16]:

```
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                            'region':['ma',np.nan,'mp','gu',np.nan],
                            'sales':[10,np.nan,30,np.nan,50],
                            'expense':[50,np.nan,70,np.nan,90]})
sales_data.dropna(subset=["sales"])
```

Out[16]:

```
   name  region  sales  expense
0     a      ma   10.0     50.0
2   NaN      mp   30.0     70.0
4     e     NaN   50.0     90.0
```

In [17]:

```
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                            'region':['ma',np.nan,'mp','gu',np.nan],
                            'sales':[10,np.nan,30,np.nan,50],
                            'expense':[50,np.nan,70,np.nan,90]})
sales_data.dropna(subset=["sales","region"])
```

Out[17]:

```
   name  region  sales  expense
0     a      ma   10.0     50.0
2   NaN      mp   30.0     70.0
```

In [18]:

```
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                            'region':['ma',np.nan,'mp','gu',np.nan],
                            'sales':[10,np.nan,30,np.nan,50],
                            'expense':[50,np.nan,70,np.nan,90]})
sales_data.dropna(thresh=3) # thresh check value atleast 3(3 or>3)
```

Out[18]:

```
   name  region  sales  expense
0     a      ma   10.0     50.0
2   NaN      mp   30.0     70.0
4     e     NaN   50.0     90.0
```

In [19]:

```
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                            'region':['ma',np.nan,'mp','gu',np.nan],
                            'sales':[10,np.nan,30,np.nan,50],
                            'expense':[50,np.nan,70,np.nan,90]})
sales_data.dropna(thresh=2) # thresh check value atleast 2(2 or>2)
```

Out[19]:

```
   name  region  sales  expense
0     a      ma   10.0     50.0
2   NaN      mp   30.0     70.0
3     d      gu    NaN      NaN
4     e     NaN   50.0     90.0
```

In [20]:

```
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                            'region':['ma',np.nan,'mp','gu',np.nan],
                            'sales':[10,np.nan,30,np.nan,50],
                            'expense':[50,np.nan,70,np.nan,90]})
sales_data.dropna(axis=0) # axis 0 = rowwise
```

Out[20]:

```
   name  region  sales  expense
0     a      ma   10.0     50.0
```

In [21]:

```python
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a','b','c','d','e'],
                            'region':['ma',np.nan,'mp','gu',np.nan],
                            'sales':[10,np.nan,30,np.nan,50],
                            'expense':[50,np.nan,70,np.nan,90]})
sales_data.dropna(axis=1) # axis 1 = columnwise
```

Out[21]:

	name
0	a
1	b
2	c
3	d
4	e

fillna()

In [22]:

```python
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                            'region':['ma',np.nan,'mp','gu',np.nan],
                            'sales':[10,np.nan,30,np.nan,50],
                            'expense':[50,np.nan,70,np.nan,90]})
sales_data.fillna(0)
```

Out[22]:

	name	region	sales	expense
0	a	ma	10.0	50.0
1	0	0	0.0	0.0
2	0	mp	30.0	70.0
3	d	gu	0.0	0.0
4	e	0	50.0	90.0

In [23]:

```python
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                            'region':['ma',np.nan,'mp','gu',np.nan],
                            'sales':[10,np.nan,30,np.nan,50],
                            'expense':[50,np.nan,70,np.nan,90]})
sales_data.fillna(sales_data["sales"].mean())
```

Out[23]:

	name	region	sales	expense
0	a	ma	10.0	50.0
1	30	30	30.0	30.0
2	30	mp	30.0	70.0
3	d	gu	30.0	30.0
4	e	30	50.0	90.0

In [24]:

```python
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                            'region':['ma',np.nan,'mp','gu',np.nan],
                            'sales':[10,np.nan,30,np.nan,50],
                            'expense':[50,np.nan,70,np.nan,90]})
sales_data['sales'].fillna(sales_data["sales"].mean())
```

Out[24]:

```
0    10.0
1    30.0
2    30.0
3    30.0
4    50.0
Name: sales, dtype: float64
```

In [25]:

```
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,np.nan,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data['sales'].fillna(sales_data["sales"].median())
```

Out[25]:

```
0    10.0
1    30.0
2    30.0
3    30.0
4    50.0
Name: sales, dtype: float64
```

In [26]:

```
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,np.nan,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data['sales'].mode()
```

Out[26]:

```
0    10.0
1    30.0
2    50.0
dtype: float64
```

In [27]:

```
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,np.nan,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data['sales'].fillna(sales_data["sales"].mode()[0])
```

Out[27]:

```
0    10.0
1    10.0
2    30.0
3    10.0
4    50.0
Name: sales, dtype: float64
```

In [28]:

```
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,np.nan,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data['sales'].fillna(sales_data["sales"].mode()[1])
```

Out[28]:

```
0    10.0
1    30.0
2    30.0
3    30.0
4    50.0
Name: sales, dtype: float64
```

In [29]:

```
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,np.nan,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data['sales'].fillna(sales_data["sales"].mode()[2])
```

Out[29]:

```
0    10.0
1    50.0
2    30.0
3    50.0
4    50.0
Name: sales, dtype: float64
```

In [30]:

```
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                            'region':['ma',np.nan,'mp','gu',np.nan],
                            'sales':[10,30,30,np.nan,50],
                            'expense':[50,np.nan,70,np.nan,90]})
sales_data
```

Out[30]:

	name	region	sales	expense
0	a	ma	10.0	50.0
1	NaN	NaN	30.0	NaN
2	NaN	mp	30.0	70.0
3	d	gu	NaN	NaN
4	e	NaN	50.0	90.0

In [31]:

```
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                            'region':['ma',np.nan,'mp','gu',np.nan],
                            'sales':[10,30,30,np.nan,50],
                            'expense':[50,np.nan,70,np.nan,90]})
sales_data['sales'].mode()
```

Out[31]:

```
0    30.0
dtype: float64
```

In [32]:

```
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                            'region':['ma',np.nan,'mp','gu',np.nan],
                            'sales':[10,30,30,np.nan,50],
                            'expense':[50,np.nan,70,np.nan,90]})
sales_data['sales'].fillna(sales_data["sales"].mode()[0])
```

Out[32]:

```
0    10.0
1    30.0
2    30.0
3    30.0
4    50.0
Name: sales, dtype: float64
```

In [33]:

```
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                            'region':['ma',np.nan,'mp','gu',np.nan],
                            'sales':[10,20,30,np.nan,50],
                            'expense':[50,np.nan,70,np.nan,90]})
sales_data['sales'].mode()
sales_data
```

Out[33]:

	name	region	sales	expense
0	a	ma	10.0	50.0
1	NaN	NaN	20.0	NaN
2	NaN	mp	30.0	70.0
3	d	gu	NaN	NaN
4	e	NaN	50.0	90.0

In [34]:

```
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                            'region':['ma',np.nan,'mp','gu',np.nan],
                            'sales':[10,20,30,np.nan,50],
                            'expense':[50,np.nan,70,np.nan,90]})
sales_data.dropna(inplace=True)
sales_data
```

Out[34]:

	name	region	sales	expense
0	a	ma	10.0	50.0

In [35]:

```python
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,20,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data = sales_data.dropna()
sales_data
```

Out[35]:

	name	region	sales	expense
0	a	ma	10.0	50.0

In [36]:

```python
import pandas as pd
import numpy as np

sales_data = pd.DataFrame({'name':['a',np.nan,np.nan,'d','e'],
                           'region':['ma',np.nan,'mp','gu',np.nan],
                           'sales':[10,20,30,np.nan,50],
                           'expense':[50,np.nan,70,np.nan,90]})
sales_data['sales'].fillna(30,inplace=True)
sales_data
```

Out[36]:

	name	region	sales	expense
0	a	ma	10.0	50.0
1	NaN	NaN	20.0	NaN
2	NaN	mp	30.0	70.0
3	d	gu	30.0	NaN
4	e	NaN	50.0	90.0

In [37]:

```python
import pandas as pd
data = {
    'A':['TA','TB','TB','TC','TA'],
    'B':[50,40,40,30,50],
    'C':[True,False,False,False,True]
}
df = pd.DataFrame(data)
dups = df.duplicated()
print(dups)
```

```
0    False
1    False
2     True
3    False
4     True
dtype: bool
```

In [38]:

```python
import pandas as pd
data = {
    'A':['TA','TB','TB','TC','TA'],
    'B':[50,40,40,30,50],
    'C':[True,False,False,False,True]
}
df = pd.DataFrame(data)
dups = df.duplicated()
print(dups)

df = df.drop_duplicates()
df
```

```
0    False
1    False
2     True
3    False
4     True
dtype: bool
```

Out[38]:

	A	B	C
0	TA	50	True
1	TB	40	False
3	TC	30	False

In [7]:

```python
import pandas as pd
data = {
    'A':['TA','TB','TB','TC','TA'],
    'B':[50,40,40,30,50],
    'C':[True,False,False,False,True]
}
df = pd.DataFrame(data)
dups = df.duplicated()
print(dups)

df = df.reset_index(drop=True)
df
```

```
0    False
1    False
2     True
3    False
4     True
dtype: bool
```

Out[7]:

	A	B	C
0	TA	50	True
1	TB	40	False
2	TB	40	False
3	TC	30	False
4	TA	50	True

In [8]:

```python
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df
```

Out[8]:

	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 [9]:

```python
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df['horsepower']=="?"
```

Out[9]:

```
0      False
1      False
2      False
3      False
4      False
       ...  
393    False
394    False
395    False
396    False
397    False
Name: horsepower, Length: 398, dtype: bool
```

In [10]:

```
import pandas as pd
import numpy as np
df = pd.read_csv('auto-mpg.csv')
df[df['horsepower']=="?"]
```

Out[10]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
32	25.0	4	98.0	?	2046	19.0	71	1	ford pinto
126	21.0	6	200.0	?	2875	17.0	74	1	ford maverick
330	40.9	4	85.0	?	1835	17.3	80	2	renault lecar deluxe
336	23.6	4	140.0	?	2905	14.3	80	1	ford mustang cobra
354	34.5	4	100.0	?	2320	15.8	81	2	renault 18i
374	23.0	4	151.0	?	3035	20.5	82	1	amc concord dl

In [13]:

```
import pandas as pd
import numpy as np

df = pd.read_csv('auto-mpg.csv')
df.loc[[32,126,330,336,354,374]]
```

Out[13]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
32	25.0	4	98.0	?	2046	19.0	71	1	ford pinto
126	21.0	6	200.0	?	2875	17.0	74	1	ford maverick
330	40.9	4	85.0	?	1835	17.3	80	2	renault lecar deluxe
336	23.6	4	140.0	?	2905	14.3	80	1	ford mustang cobra
354	34.5	4	100.0	?	2320	15.8	81	2	renault 18i
374	23.0	4	151.0	?	3035	20.5	82	1	amc concord dl

In [15]:

```
import pandas as pd
import numpy as np

df = pd.read_csv('auto-mpg.csv')
df[df['horsepower']!="?"]
```

Out[15]:

	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

392 rows × 9 columns

In [16]:

```python
import pandas as pd
import numpy as np

df = pd.read_csv('auto-mpg.csv')
df.drop('mpg',axis=1) # axis 1 column delete.
```

Out[16]:

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

398 rows × 8 columns

In [17]:

```python
import pandas as pd
import numpy as np

df = pd.read_csv('auto-mpg.csv')
df.drop(2,axis=0) # 2 nd row will be delete.
```

Out[17]:

	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
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
...	...	...	...	...	...	...	...	...	...
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

397 rows × 9 columns

Outlier :

In [25]:

```python
import pandas as pd
import numpy as np

def find_outlier(ds,col):
    Q1 = ds[col].quantile(0.25)
    Q3 = ds[col].quantile(0.75)
    IQR = Q3 - Q1
    low_val = Q1 - (1.5*IQR)
    high_val = Q3 + (1.5*IQR)
    ds = ds.loc[(ds[col]<low_val)|(ds[col]>high_val)]
    return ds

df = pd.read_csv('auto-mpg.csv')
find_outlier(df,"mpg")
```

Out[25]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
322	46.6	4	86.0	65	2110	17.9	80	3	mazda glc

In [24]:

```
import pandas as pd
import numpy as np

df = pd.read_csv('auto-mpg.csv')
df.describe()
```

Out[24]:

	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 [26]:

```
import pandas as pd
import numpy as np

def find_outlier(ds,col):
    Q1 = ds[col].quantile(0.25)
    Q3 = ds[col].quantile(0.75)
    IQR = Q3 - Q1
    low_val = Q1 - (1.5*IQR)
    high_val = Q3 + (1.5*IQR)
    ds = ds.loc[(ds[col]<low_val)|(ds[col]>high_val)]
    return ds

df = pd.read_csv('auto-mpg.csv')
find_outlier(df,"acceleration")
```

Out[26]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
7	14.0	8	440.0	215	4312	8.5	70	1	plymouth fury iii
9	15.0	8	390.0	190	3850	8.5	70	1	amc ambassador dpl
11	14.0	8	340.0	160	3609	8.0	70	1	plymouth 'cuda 340
59	23.0	4	97.0	54	2254	23.5	72	2	volkswagen type 3
195	29.0	4	85.0	52	2035	22.2	76	1	chevrolet chevette
299	27.2	4	141.0	71	3190	24.8	79	2	peugeot 504
300	23.9	8	260.0	90	3420	22.2	79	1	oldsmobile cutlass salon brougham
326	43.4	4	90.0	48	2335	23.7	80	2	vw dasher (diesel)
394	44.0	4	97.0	52	2130	24.6	82	2	vw pickup

In [27]:

```
import pandas as pd
import numpy as np

def find_outlier(ds,col):
    Q1 = ds[col].quantile(0.25)
    Q3 = ds[col].quantile(0.75)
    IQR = Q3 - Q1
    low_val = Q1 - (1.5*IQR)
    high_val = Q3 + (1.5*IQR)
    ds = ds.loc[(ds[col]>low_val)&(ds[col]<high_val)]
    return ds

df = pd.read_csv('auto-mpg.csv')
find_outlier(df,"mpg")
```

Out[27]:

	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

397 rows × 9 columns

In [29]:

```
import pandas as pd
import numpy as np

def find_outlier(ds,col):
    Q1 = ds[col].quantile(0.25)
    Q3 = ds[col].quantile(0.75)
    IQR = Q3 - Q1
    low_val = Q1 - (1.5*IQR)
    high_val = Q3 + (1.5*IQR)
    ds = ds.loc[(ds[col]>low_val)&(ds[col]<high_val)]
    return ds

df = pd.read_csv('auto-mpg.csv')
find_outlier(df,"acceleration")
```

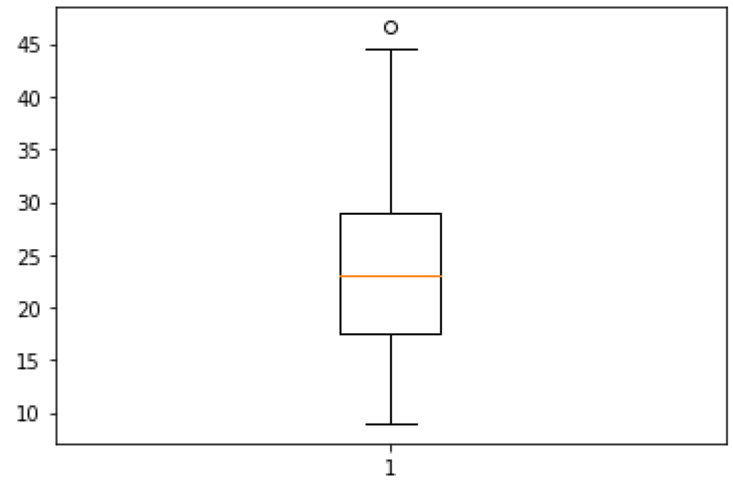
Out[29]:

	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
...	...	...	...	...	...	...	...	...	...
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
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

389 rows × 9 columns

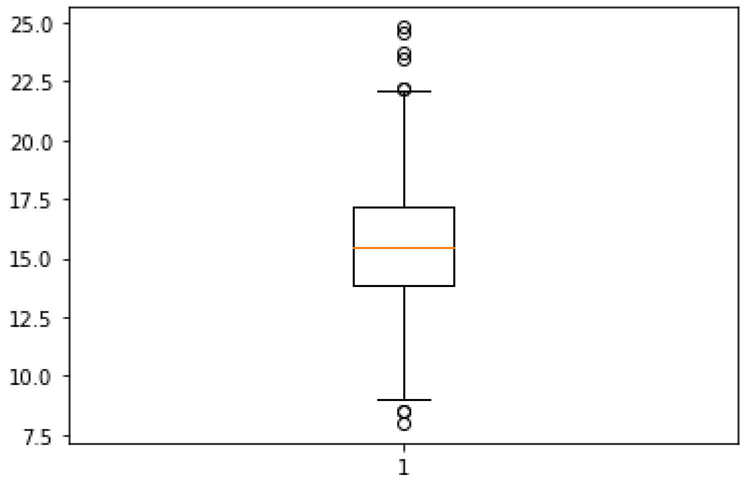
In [31]:

```python
import matplotlib.pyplot as plt
plt.boxplot(df["mpg"])
plt.show()
```



In [32]:

```python
import matplotlib.pyplot as plt
plt.boxplot(df["acceleration"])
plt.show()
```



In [ ]: