In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
df=pd.read_csv(r'E:\python\2. Cars Data1.csv')
```

In [3]:
```
df
```

3	Acura	TL 4dr	Sedan	Asia	Front	$33,195	$30,299	3.2	6.0
4	Acura	3.5 RL 4dr	Sedan	Asia	Front	$43,755	$39,014	3.5	6.0
...	...	...	...	...	...	...	...	...	...
427	Volvo	C70 LPT convertible 2dr	Sedan	Europe	Front	$40,565	$38,203	2.4	5.0
428	Volvo	C70 HPT convertible 2dr	Sedan	Europe	Front	$42,565	$40,083	2.3	5.0
429	Volvo	S80 T6 4dr	Sedan	Europe	Front	$45,210	$42,573	2.9	6.0
430	Volvo	V40	Wagon	Europe	Front	$26,135	$24,641	1.9	4.0
431	Volvo	XC70	Wagon	Europe	All	$35,145	$33,112	2.5	5.0

432 rows × 15 columns

In [4]:
```
df.head()   # to see first 5 record
```

Out[4]:

	Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize	Cylinders	Horsepower
0	Acura	MDX	SUV	Asia	All	$36,945	$33,337	3.5	6.0	265.0
1	Acura	RSX Type S 2dr	Sedan	Asia	Front	$23,820	$21,761	2.0	4.0	200.0
2	Acura	TSX 4dr	Sedan	Asia	Front	$26,990	$24,647	2.4	4.0	200.0
3	Acura	TL 4dr	Sedan	Asia	Front	$33,195	$30,299	3.2	6.0	270.0
4	Acura	3.5 RL 4dr	Sedan	Asia	Front	$43,755	$39,014	3.5	6.0	225.0

In [5]:
```
df.info()        # information regarding data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 432 entries, 0 to 431
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Make         428 non-null    object 
 1   Model        428 non-null    object 
 2   Type         428 non-null    object 
 3   Origin       428 non-null    object 
 4   DriveTrain   428 non-null    object 
 5   MSRP         428 non-null    object 
 6   Invoice      428 non-null    object 
 7   EngineSize   428 non-null    float64
 8   Cylinders    426 non-null    float64
 9   Horsepower   428 non-null    float64
 10  MPG_City     428 non-null    float64
 11  MPG_Highway  428 non-null    float64
 12  Weight       428 non-null    float64
 13  Wheelbase    428 non-null    float64
 14  Length       428 non-null    float64
dtypes: float64(8), object(7)
memory usage: 50.8+ KB
```

In [6]:
```
df.describe()
```

Out[6]:

	EngineSize	Cylinders	Horsepower	MPG_City	MPG_Highway	Weight	Wheelbase
count	428.000000	426.000000	428.000000	428.000000	428.000000	428.000000	428.000000
mean	3.196729	5.807512	215.885514	20.060748	26.843458	3577.953271	108.154206
std	1.108595	1.558443	71.836032	5.238218	5.741201	758.983215	8.311813
min	1.300000	3.000000	73.000000	10.000000	12.000000	1850.000000	89.000000
25%	2.375000	4.000000	165.000000	17.000000	24.000000	3104.000000	103.000000
50%	3.000000	6.000000	210.000000	19.000000	26.000000	3474.500000	107.000000
75%	3.900000	6.000000	255.000000	21.250000	29.000000	3977.750000	112.000000
max	8.300000	12.000000	500.000000	60.000000	66.000000	7190.000000	144.000000

In [7]:
```
# get the unique catgory counts
df['Origin'].value_counts()
```

Out[7]:
```
Asia      158
USA       147
Europe    123
Name: Origin, dtype: int64
```

In [8]:
```
df[df['Cylinders']>5]
```

Out[8]:

	Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize	Cylinders
0	Acura	MDX	SUV	Asia	All	$36,945	$33,337	3.5	6.(
3	Acura	TL 4dr	Sedan	Asia	Front	$33,195	$30,299	3.2	6.(
4	Acura	3.5 RL 4dr	Sedan	Asia	Front	$43,755	$39,014	3.5	6.(
5	Acura	3.5 RL w/Navigation 4dr	Sedan	Asia	Front	$46,100	$41,100	3.5	6.(
6	Acura	NSX coupe 2dr manual S	Sports	Asia	Rear	$89,765	$79,978	3.2	6.(
...	...	...	...	...	...	...	...	...	..
416	Volkswagen	Phaeton W12 4dr	Sedan	Europe	Front	$75,000	$69,130	6.0	12.(
419	Volkswagen	Passat W8	Wagon	Europe	Front	$40,235	$36,956	4.0	8.(
420	Volvo	XC90 T6	SUV	Europe	All	$41,250	$38,851	2.9	6.(
425	Volvo	S80 2.9 4dr	Sedan	Europe	Front	$37,730	$35,542	2.9	6.(
429	Volvo	S80 T6 4dr	Sedan	Europe	Front	$45,210	$42,573	2.9	6.(

282 rows × 15 columns

CSV

In [9]:
```
from io import StringIO, BytesIO
```

In [10]:
```
data = ('col1,col2,col3\n'
           'x,y,1\n'
        'a,b,2\n'
        'c,d,3')                    # \n for next line
```

In [11]:
```
type(data)
```

Out[11]:
```
str
```

```
stringIO()
```

In [12]:
```
StringIO()
```

Out[12]:
```
<_io.StringIO at 0x1c4913633a0>
```

In [13]:
```
pd.read_csv(StringIO(data))
```

Out[13]:

	col1	col2	col3
0	x	y	1
1	a	b	2
2	c	d	3

In [14]:
```
## read from specific columns
df=pd.read_csv(StringIO(data), usecols=['col1','col3'])
```

In [15]:
```
df
```

Out[15]:

	col1	col3
0	x	1
1	a	2
2	c	3

In [16]:
```
df=pd.read_csv(StringIO(data))
```

In [17]:
```
df.to_csv('test.csv')
```

In [83]:
```
## specifying columns data types
data=('a,b,c,d\n'
            '1,2,3,4\n'
            '5,6,7,8\n'
            '9,10,11,12')
```

In [84]:
```
print(data)
```

```
a,b,c,d
1,2,3,4
5,6,7,8
9,10,11,12
```

In [92]:
```
df=pd.read_csv(StringIO(data),dtype=float)
```

In [93]:
```
df
```

Out[93]:

	a	b	c	d
0	1.0	2.0	3.0	4.0
1	5.0	6.0	7.0	8.0
2	9.0	10.0	11.0	12.0

In [ ]:
```
```

In [94]:
```
df['a'][1]
```

Out[94]:
```
5.0
```

In [95]:
```
df['a']
```

Out[95]:
```
0    1.0
1    5.0
2    9.0
Name: a, dtype: float64
```

In [96]:
```
df=pd.read_csv(StringIO(data),dtype=float)
```

In [97]:
```
df
```

Out[97]:

	a	b	c	d
0	1.0	2.0	3.0	4.0
1	5.0	6.0	7.0	8.0
2	9.0	10.0	11.0	12.0

In [98]:
```
df=pd.read_csv(StringIO(data),dtype={'b':int,'c':float,'a':'Int64'})
```

In [99]:
```
df
```

Out[99]:

	a	b	c	d
0	1	2	3.0	4
1	5	6	7.0	8
2	9	10	11.0	12

In [105]:
```
type(df['c'][1])
```

Out[105]:
```
numpy.float64
```

In [101]:
```
## check the datatype
df.dtypes
```

Out[101]:
```
a      Int64
b      int32
c    float64
d      int64
dtype: object
```

In [106]:
```
data1=('index,a,b,c\n'
       '4,apple,bat,5.7\n'
        '8,orange,cow,10')
```

In [103]:
```
data1
```

Out[103]:
```
'index,a,b,c\n4,apple,bat,5.7\n8,orange,cow,10'
```

In [108]:
```
pd.read_csv(StringIO(data1))
```

Out[108]:

	index	a	b	c
0	4	apple	bat	5.7
1	8	orange	cow	10.0

In [110]:
```
pd.read_csv(StringIO(data1),index_col=1)  # 0,1,2... use for making index colu
```

Out[110]:

	index	b	c
a			
apple	4	bat	5.7
orange	8	cow	10.0

In [113]:
```
data=('a,b,c\n'
     '4,apple,bat,\n'
     '8,orange,cow,')
```

In [116]:
```
pd.read_csv(StringIO(data))
```

Out[116]:

	a	b	c
4	apple	bat	NaN
8	orange	cow	NaN

In [117]:
```
pd.read_csv(StringIO(data),index_col=False)
```

Out[117]:

	a	b	c
0	4	apple	bat
1	8	orange	cow

In [118]:
```
## combining usecols and index_col
data=('a,b,c\n'
     '4,apple,bat,\n'
     '8,orange,cow,')
```

In [119]:
```
pd.read_csv(StringIO(data),usecols=['b','c'],index_col=False)
```

Out[119]:

	b	c
0	apple	bat
1	orange	cow

In [128]:
```
## quoting and Escape characters very useful in NLP
data='a,b\n"hello,\\"Bob\\", nice to see you",5'
```

In [129]:
```
data
```

Out[129]:
```
'a,b\n"hello,\\"Bob\\", nice to see you",5'
```

In [130]:
```
pd.read_csv(StringIO(data),escapechar='\\')
```

Out[130]:

	a	b
0	hello,"Bob", nice to see you	5

In [131]:
```
## URL to CSV
df=pd.read_csv('https://download.bls.gov/pub/time.series/cu/cu.item',sep='\t')
```

```
---------------------------------------------------------------------------
gaierror                                  Traceback (most recent call last)
File E:\anaconda\lib\urllib\request.py:1346, in AbstractHTTPHandler.do_open
(self, http_class, req, **http_conn_args)
   1345 try:
-> 1346     h.request(req.get_method(), req.selector, req.data, headers,
   1347               encode_chunked=req.has_header('Transfer-encoding'))
   1348 except OSError as err: # timeout error

File E:\anaconda\lib\http\client.py:1285, in HTTPConnection.request(self, m
ethod, url, body, headers, encode_chunked)
   1284 """Send a complete request to the server."""
-> 1285 self._send_request(method, url, body, headers, encode_chunked)

File E:\anaconda\lib\http\client.py:1331, in HTTPConnection._send_request(s
elf, method, url, body, headers, encode_chunked)
   1330     body = _encode(body, 'body')
-> 1331 self.endheaders(body, encode_chunked=encode_chunked)

File E:\anaconda\lib\http\client.py:1280, in HTTPConnection.endheaders(sel
```

In [ ]: