In [43]:
```
import pandas as pd
pd.set_option('display.max_columns', None)  # 'None' mean that we want to display all rows and columns,
pd.set_option('display.max_rows', None) #we can specify the number we want to display
```

In [44]:
```
#read the csv file and specify the id as an index
df=pd.read_csv('MOCK_DATA.csv',index_col='id')
df
```

Out[44]:

id	first_name	last_name	email	gender	ip_address
1	Pearl	Ennals	pennals0@google.ru	Female	254.38.75.30
2	Nigel	Kelland	nkelland1@1688.com	Male	59.136.94.67
3	Rolando	Gammage	rgammage2@dyndns.org	Male	251.16.66.148
4	Cullen	Hamon	chamon3@cafepress.com	Male	45.28.192.77
5	Caesar	Raggitt	craggitt4@kickstarter.com	Male	97.16.70.243
...	...	...	...	...	...
996	Prisca	Ellington	pellingtonrm@ehow.com	Female	53.205.195.7
997	Rosalinda	Grut	rgrutro@squarespace.com	Female	195.44.160.243
998	Melantha	Haldane	mhaldanerp@ftc.gov	Female	231.48.191.112
999	Hilton	Marzelle	hmarzellerq@creativecommons.org	Male	119.12.216.126
1000	Lisetta	Shoard	lshoardrr@time.com	Female	91.45.172.114

1000 rows × 5 columns

In [34]:
```
#return the columns in our dataframe
df.columns
```

Out[34]:
```
Index(['first_name', 'last_name', 'email', 'gender', 'ip_address'], dtype='object')
```

In [35]:
```
#by default the head fuction display the first 5 rows, and we can specify the number into the brackets
df.head(10)
```

Out[35]:

id	first_name	last_name	email	gender	ip_address
1	Pearl	Ennals	pennals0@google.ru	Female	254.38.75.30
2	Nigel	Kelland	nkelland1@1688.com	Male	59.136.94.67
3	Rolando	Gammage	rgammage2@dyndns.org	Male	251.16.66.148
4	Cullen	Hamon	chamon3@cafepress.com	Male	45.28.192.77
5	Caesar	Raggitt	craggitt4@kickstarter.com	Male	97.16.70.243
6	Jeff	Rosiello	jrosiello5@drupal.org	Male	63.106.205.87
7	Gerti	Quinell	gquinell6@youku.com	Female	18.121.106.31
8	Carlye	Constanza	cconstanza7@census.gov	Female	102.166.197.235
9	Hazel	Olden	holden8@netvibes.com	Male	219.53.144.66
10	Wolfie	Cantor	wcantor9@japanpost.jp	Male	108.251.205.65

In [12]:
```
#by default display the last 5 rows, and we can specify the number into the brackets
df.tail(2)
```

Out[12]:

id	first_name	last_name	email	gender	ip_address
999	Hilton	Marzelle	hmarzellerq@creativecommons.org	Male	119.12.216.126
1000	Lisetta	Shoard	lshoardrr@time.com	Female	91.45.172.114

In [13]:
```
#return the dimension of the DataFrame
df.shape
```

Out[13]:
```
(1000, 5)
```

In [25]:
```
#display a specific column exemple 'last_name'
df['last_name']
```

Out[25]:
```
id
1         Ennals
2        Kelland
          ...
999      Marzelle
1000      Shoard
Name: last_name, Length: 1000, dtype: object
```

In [26]:
```
#loc is used to specify both row and column with label
#in this exemple we want to display the row number 2
df.loc[2]
```

Out[26]:
```
first_name          Nigel
last_name          Kelland
                    ...
gender                Male
ip_address     59.136.94.67
Name: 2, Length: 5, dtype: object
```

In [27]:
```
#display rows from row 1 to row 10
df.loc[1:10]
```

Out[27]:

id	first_name	last_name	...	gender	ip_address
1	Pearl	Ennals	...	Female	254.38.75.30
2	Nigel	Kelland	...	Male	59.136.94.67
...	...	...	...	...	...
9	Hazel	Olden	...	Male	219.53.144.66
10	Wolfie	Cantor	...	Male	108.251.205.65

10 rows × 5 columns

In [28]:
```
#also we can specifiy wich column we want to display, in this exemple we want to display cols from emailt to ip_address
df.loc[1:10,'email':'ip_address']
```

Out[28]:

id	email	gender	ip_address
1	pennals0@google.ru	Female	254.38.75.30
2	nkelland1@1688.com	Male	59.136.94.67
...	...	...	...
9	holden8@netvibes.com	Male	219.53.144.66
10	wcantor9@japanpost.jp	Male	108.251.205.65

10 rows × 3 columns

In [45]:
```
#to access multiple lines use double brackets separated by commas and specify the labels
df [['email', 'gender']]
df.head(15)
```

Out[45]:

id	first_name	last_name	email	gender	ip_address
1	Pearl	Ennals	pennals0@google.ru	Female	254.38.75.30
2	Nigel	Kelland	nkelland1@1688.com	Male	59.136.94.67
3	Rolando	Gammage	rgammage2@dyndns.org	Male	251.16.66.148
4	Cullen	Hamon	chamon3@cafepress.com	Male	45.28.192.77
5	Caesar	Raggitt	craggitt4@kickstarter.com	Male	97.16.70.243
6	Jeff	Rosiello	jrosiello5@drupal.org	Male	63.106.205.87
7	Gerti	Quinell	gquinell6@youku.com	Female	18.121.106.31
8	Carlye	Constanza	cconstanza7@census.gov	Female	102.166.197.235
9	Hazel	Olden	holden8@netvibes.com	Male	219.53.144.66
10	Wolfie	Cantor	wcantor9@japanpost.jp	Male	108.251.205.65
11	Eldin	Allnutt	eallnutta@sciencedirect.com	Male	235.57.81.1
12	Oneida	Moss	omossb@behance.net	Female	60.10.182.80
13	Tiebold	Dresselc	tdresselc@washington.edu	Male	118.166.242.92
14	Olivero	Worvill	ovorvillis@senate.gov	Male	63.86.106.254
15	Orion	Mickleburgh	omickleburghe@drupal.org	Male	157.25.105.27

In [35]:
```
#iloc allows to access row by location (position)
df.iloc[1]
```

Out[35]:
```
first_name          Nigel
last_name          Kelland
                    ...
gender                Male
ip_address     59.136.94.67
Name: 2, Length: 5, dtype: object
```

In [36]:
```
df.iloc[[0,1]]
```

Out[36]:

id	first_name	last_name	...	gender	ip_address
1	Pearl	Ennals	...	Female	254.38.75.30
2	Nigel	Kelland	...	Male	59.136.94.67

2 rows × 5 columns

In [37]:
```
#we can separate rows and coloumns with the double brakets, the first brakets[0,1] means rows with index 0 and 1,
#then the column with index 1 wich is 'last name'
df.iloc[[0,1],1]
```

Out[37]:
```
id
1    Ennals
2    Kelland
Name: last_name, dtype: object
```

In [38]:
```
#here we can get the same result by using loc[] with labels
df.loc[0:2,'last_name']
```

Out[38]:
```
id
1    Ennals
2    Kelland
Name: last_name, dtype: object
```

In [39]:
```
#Reset the index of the DataFrame wich is the id , and use the default one instead.
df.reset_index(inplace=True)
```

In [40]:
```
df.loc[[0,1,2], ['last_name','gender']]
```

Out[40]:

	last_name	gender
0	Ennals	Female
1	Kelland	Male
2	Gammage	Male

In [41]:
```
#Sort the index descending
df.sort_index(ascending=False)
```

Out[41]:

	id	first_name	...	gender	ip_address
999	1000	Lisetta	...	Female	91.45.172.114
998	999	Hilton	...	Male	119.12.216.126
...	...	...	...	...	...
1	2	Nigel	...	Male	59.136.94.67
0	1	Pearl	...	Female	254.38.75.30

1000 rows × 6 columns

In [43]:
```
#set the column headers to uppercase
df.columns=[x.upper() for x in df.columns]
df.head()
```

Out[43]:

	ID	FIRST_NAME	...	GENDER	IP_ADDRESS
0	1	Pearl	...	Female	254.38.75.30
1	2	Nigel	...	Male	59.136.94.67
...	...	...	...	...	...
3	4	Cullen	...	Male	45.28.192.77
4	5	Caesar	...	Male	97.16.70.243

5 rows × 6 columns

In [45]:
```
#set the column headers to lowercase
df.columns=[x.lower() for x in df.columns]
df.head()
```

Out[45]:

	id	first_name	...	gender	ip_address
0	1	Pearl	...	Female	254.38.75.30
1	2	Nigel	...	Male	59.136.94.67
...	...	...	...	...	...
3	4	Cullen	...	Male	45.28.192.77
4	5	Caesar	...	Male	97.16.70.243

5 rows × 6 columns