```
import numpy as np
import pandas as pd


# CONSIDERIAMO IL NOSTRO SOLITO Df tips E VEDIAMO QUALI METODI
POSSIAMO APPLICARE AD ESSO (E,
# IN GENERALE, A TUTTI I DATAFRAMES


tips = pd.read_csv("tips.csv")
tips.head()
```

```
   total_bill   tip     sex smoker  day    time  size
price_per_person  \
0       16.99  1.01  Female     No  Sun  Dinner     2
8.49
1       10.34  1.66    Male     No  Sun  Dinner     3
3.45
2       21.01  3.50    Male     No  Sun  Dinner     3
7.00
3       23.68  3.31    Male     No  Sun  Dinner     2
11.84
4       24.59  3.61  Female     No  Sun  Dinner     4
6.15

           Payer Name         CC Number Payment ID
0  Christy Cunningham  3560325168603410    Sun2959
1      Douglas Tucker  4478071379779230    Sun4608
2      Travis Walters  6011812112971322    Sun4458
3    Nathaniel Harris  4676137647685994    Sun5260
4        Tonya Carter  4832732618637221    Sun2251
```

```
tips.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   total_bill        244 non-null    float64
 1   tip               244 non-null    float64
 2   sex               244 non-null    object
 3   smoker            244 non-null    object
 4   day               244 non-null    object
 5   time              244 non-null    object
 6   size              244 non-null    int64
 7   price_per_person  244 non-null    float64
```

```
 8   Payer Name        244 non-null    object
 9   CC Number         244 non-null    int64
 10  Payment ID        244 non-null    object
dtypes: float64(3), int64(2), object(6)
memory usage: 21.1+ KB
```

```
# tips.info() è molto utile per fare una prima pulizia dei dati.
Infatti, spesso,
#nei df scaricati da internet le tipologie delle colonne sono errate.
```

```
# Supponiamo di voler calcolare le statistiche di base: media, moda,
mediana etc. Possiamo vederle tramite
# la funzione describe(). Cosa succede se la applichiamo?
```

```
tips.describe()
```

```
       total_bill         tip        size  price_per_person     CC
Number
count  244.000000  244.000000  244.000000        244.000000
2.440000e+02
mean    19.785943    2.998279    2.569672          7.888197
2.563496e+15
std      8.902412    1.383638    0.951100          2.914234
2.369340e+15
min      3.070000    1.000000    1.000000          2.880000
6.040679e+10
25%     13.347500    2.000000    2.000000          5.800000
3.040731e+13
50%     17.795000    2.900000    2.000000          7.255000
3.525318e+15
75%     24.127500    3.562500    3.000000          9.390000
4.553675e+15
max     50.810000   10.000000    6.000000         20.270000
6.596454e+15
```

```
# Notiamo subito che queste statistiche sono calcolate anche sul CC
number. Questo non ha senso.
# Nonostante tutti gli elementi sono numeri, in realtà dovremmo
castarlo a str. Ripartiamo quindi dal
# nostro Df.info
```

```
tips.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   total_bill        244 non-null    float64
 1   tip               244 non-null    float64
 2   sex               244 non-null    object 
 3   smoker            244 non-null    object 
 4   day               244 non-null    object 
 5   time              244 non-null    object 
 6   size              244 non-null    int64  
 7   price_per_person  244 non-null    float64
 8   Payer Name        244 non-null    object 
 9   CC Number         244 non-null    int64  
 10  Payment ID        244 non-null    object 
dtypes: float64(3), int64(2), object(6)
memory usage: 21.1+ KB
```

```
# CC NUMBER DEVE DIVENTARE str. Per farlo, si utilizza il metodo astype:

tips['CC Number'] = tips['CC Number'].astype('str')
```

```
tips.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   total_bill        244 non-null    float64
 1   tip               244 non-null    float64
 2   sex               244 non-null    object 
 3   smoker            244 non-null    object 
 4   day               244 non-null    object 
 5   time              244 non-null    object 
 6   size              244 non-null    int64  
 7   price_per_person  244 non-null    float64
 8   Payer Name        244 non-null    object 
 9   CC Number         244 non-null    object 
 10  Payment ID        244 non-null    object 
dtypes: float64(3), int64(1), object(7)
memory usage: 21.1+ KB
```

```
# In Pandas, le stringhe sono oggetti => CC Number è diventato un
object.


# Possiamo quindi ora calcolare le statistiche di base

tips.describe().round(2)
```

```
       total_bill     tip    size  price_per_person
count      244.00  244.00  244.00            244.00
mean        19.79    3.00    2.57              7.89
std          8.90    1.38    0.95              2.91
min          3.07    1.00    1.00              2.88
25%         13.35    2.00    2.00              5.80
50%         17.80    2.90    2.00              7.26
75%         24.13    3.56    3.00              9.39
max         50.81   10.00    6.00             20.27
```

```
# A volte è più facile vedere la trasposta

tips.describe().transpose().round(2)
```

```
                  count   mean   std   min    25%    50%    75%    max
total_bill        244.0  19.79  8.90  3.07  13.35  17.80  24.13  50.81
tip               244.0   3.00  1.38  1.00   2.00   2.90   3.56  10.00
size              244.0   2.57  0.95  1.00   2.00   2.00   3.00   6.00
price_per_person  244.0   7.89  2.91  2.88   5.80   7.26   9.39  20.27
```

```
tips = pd.read_csv("tips.csv")
tips.head()
```

```
   total_bill   tip     sex smoker  day    time  size
price_per_person  \
0       16.99  1.01  Female     No  Sun  Dinner     2
8.49
1       10.34  1.66    Male     No  Sun  Dinner     3
3.45
2       21.01  3.50    Male     No  Sun  Dinner     3
7.00
3       23.68  3.31    Male     No  Sun  Dinner     2
11.84
4       24.59  3.61  Female     No  Sun  Dinner     4
6.15

           Payer Name          CC Number Payment ID
0  Christy Cunningham  3560325168603410    Sun2959
1      Douglas Tucker  4478071379779230    Sun4608
```

```
2      Travis Walters  6011812112971322    Sun4458
3    Nathaniel Harris  4676137647685994    Sun5260
4        Tonya Carter  4832732618637221    Sun2251

# Ordinare le righe in base ad una colonna: sort_values()

# Supponiamo di voler ordinare il Df in base al price_per_person in
ordine crescente.


tips.sort_values(by='price_per_person')

     total_bill    tip     sex smoker  day    time  size
price_per_person  \
92         5.75   1.00  Female    Yes  Fri  Dinner     2
2.88
67         3.07   1.00  Female    Yes  Sat  Dinner     1
3.07
16        10.33   1.67  Female     No  Sun  Dinner     3
3.44
1         10.34   1.66    Male     No  Sun  Dinner     3
3.45
172        7.25   5.15    Male    Yes  Sun  Dinner     2
3.62
..          ...    ...     ...    ...  ...     ...   ...
...
237       32.83   1.17    Male    Yes  Sat  Dinner     2
16.42
175       32.90   3.11    Male    Yes  Sun  Dinner     2
16.45
170       50.81  10.00    Male    Yes  Sat  Dinner     3
16.94
179       34.63   3.55    Male    Yes  Sun  Dinner     2
17.32
184       40.55   3.00    Male    Yes  Sun  Dinner     2
20.27

           Payer Name         CC Number Payment ID
92       Leah Ramirez  3508911676966392    Fri3780
67      Tiffany Brock  4359488526995267    Sat3455
16   Elizabeth Foster  4240025044626033    Sun9715
1      Douglas Tucker  4478071379779230    Sun4608
172       Larry White    30432617123103    Sun9209
..                ...               ...        ...
237      Thomas Brown  4284722681265508    Sat2929
175   Nathan Reynolds   370307040837149    Sun5109
170     Gregory Clark  5473850968388236    Sat1954
179      Brian Bailey   346656312114848    Sun9851
184       Stephen Cox  3547798222044029    Sun5140
```

```
[244 rows x 11 columns]

tips

     total_bill   tip     sex smoker   day    time  size
price_per_person  \
0         16.99  1.01  Female     No   Sun  Dinner     2
8.49
1         10.34  1.66    Male     No   Sun  Dinner     3
3.45
2         21.01  3.50    Male     No   Sun  Dinner     3
7.00
3         23.68  3.31    Male     No   Sun  Dinner     2
11.84
4         24.59  3.61  Female     No   Sun  Dinner     4
6.15
..          ...   ...     ...    ...   ...     ...   ...
...
239       29.03  5.92    Male     No   Sat  Dinner     3
9.68
240       27.18  2.00  Female    Yes   Sat  Dinner     2
13.59
241       22.67  2.00    Male    Yes   Sat  Dinner     2
11.34
242       17.82  1.75    Male     No   Sat  Dinner     2
8.91
243       18.78  3.00  Female     No  Thur  Dinner     2
9.39

             Payer Name         CC Number Payment ID
0    Christy Cunningham  3560325168603410    Sun2959
1        Douglas Tucker  4478071379779230    Sun4608
2        Travis Walters  6011812112971322    Sun4458
3      Nathaniel Harris  4676137647685994    Sun5260
4          Tonya Carter  4832732618637221    Sun2251
..                  ...               ...        ...
239       Michael Avila  5296068606052842    Sat2657
240      Monica Sanders  3506806155565404    Sat1766
241          Keith Wong  6011891618747196    Sat3880
242        Dennis Dixon     4375220550950      Sat17
243     Michelle Hardin  3511451626698139    Thur672

[244 rows x 11 columns]


# Ovviamente il Df di partenza non presenta l'ordinamento cambiato.
Per cambiarlo:
```

```
tips =tips.sort_values(by='price_per_person')
tips
```

```
     total_bill    tip     sex smoker  day    time  size
price_per_person  \
92         5.75   1.00  Female    Yes  Fri  Dinner     2
2.88
67         3.07   1.00  Female    Yes  Sat  Dinner     1
3.07
16        10.33   1.67  Female     No  Sun  Dinner     3
3.44
1         10.34   1.66    Male     No  Sun  Dinner     3
3.45
172        7.25   5.15    Male    Yes  Sun  Dinner     2
3.62
..          ...    ...     ...    ...  ...     ...   ...
...
237       32.83   1.17    Male    Yes  Sat  Dinner     2
16.42
175       32.90   3.11    Male    Yes  Sun  Dinner     2
16.45
170       50.81  10.00    Male    Yes  Sat  Dinner     3
16.94
179       34.63   3.55    Male    Yes  Sun  Dinner     2
17.32
184       40.55   3.00    Male    Yes  Sun  Dinner     2
20.27

            Payer Name         CC Number Payment ID
92       Leah Ramirez  3508911676966392    Fri3780
67      Tiffany Brock  4359488526995267    Sat3455
16   Elizabeth Foster  4240025044626033    Sun9715
1      Douglas Tucker  4478071379779230    Sun4608
172       Larry White    30432617123103    Sun9209
..                ...               ...        ...
237      Thomas Brown  4284722681265508    Sat2929
175   Nathan Reynolds   370307040837149    Sun5109
170     Gregory Clark  5473850968388236    Sat1954
179      Brian Bailey   346656312114848    Sun9851
184       Stephen Cox  3547798222044029    Sun5140

[244 rows x 11 columns]
```

```
# CVD


# Supponiamo ora di voler ordinare il Df in base al price_per_person
in ordine decrescente.
```

```
tips = tips.sort_values(by="price_per_person",ascending=False)
tips
```

```
     total_bill    tip     sex smoker  day    time  size
price_per_person  \
184       40.55   3.00    Male    Yes  Sun  Dinner     2
20.27
179       34.63   3.55    Male    Yes  Sun  Dinner     2
17.32
170       50.81  10.00    Male    Yes  Sat  Dinner     3
16.94
175       32.90   3.11    Male    Yes  Sun  Dinner     2
16.45
237       32.83   1.17    Male    Yes  Sat  Dinner     2
16.42
..          ...    ...     ...    ...  ...     ...   ...
...
172        7.25   5.15    Male    Yes  Sun  Dinner     2
3.62
1         10.34   1.66    Male     No  Sun  Dinner     3
3.45
16        10.33   1.67  Female     No  Sun  Dinner     3
3.44
67         3.07   1.00  Female    Yes  Sat  Dinner     1
3.07
92         5.75   1.00  Female    Yes  Fri  Dinner     2
2.88

           Payer Name         CC Number Payment ID
184       Stephen Cox  3547798222044029    Sun5140
179      Brian Bailey   346656312114848    Sun9851
170     Gregory Clark  5473850968388236    Sat1954
175   Nathan Reynolds   370307040837149    Sun5109
237      Thomas Brown  4284722681265508    Sat2929
..                ...               ...        ...
172       Larry White    30432617123103    Sun9209
1      Douglas Tucker  4478071379779230    Sun4608
16   Elizabeth Foster  4240025044626033    Sun9715
67      Tiffany Brock  4359488526995267    Sat3455
92       Leah Ramirez  3508911676966392    Fri3780

[244 rows x 11 columns]
```

```
# CVD
```

```
# Possiamo ordinare il Df anche in base a più colonne. Consideriamo la variabile day, che presenta 4 valori possibili.
# Dato che è una variabile str (object), l'ordinamento ordinerà (Che brutto italiano!) la colonna in ordine alfabetico


tips = tips.sort_values(by="day")
tips
```

```
     total_bill   tip     sex smoker   day    time  size
price_per_person  \
92         5.75  1.00  Female    Yes   Fri  Dinner     2
2.88
101       15.38  3.00  Female    Yes   Fri  Dinner     2
7.69
99        12.46  1.50    Male     No   Fri  Dinner     2
6.23
220       12.16  2.20    Male    Yes   Fri   Lunch     2
6.08
97        12.03  1.50    Male    Yes   Fri  Dinner     2
6.02
..          ...   ...     ...    ...   ...     ...   ...
...
144       16.43  2.30  Female     No  Thur   Lunch     2
8.22
203       16.40  2.50  Female    Yes  Thur   Lunch     2
8.20
119       24.08  2.92  Female     No  Thur   Lunch     4
6.02
81        16.66  3.40    Male     No  Thur   Lunch     2
8.33
127       14.52  2.00  Female     No  Thur   Lunch     2
7.26

          Payer Name         CC Number Payment ID
92      Leah Ramirez  3508911676966392    Fri3780
101    Tiffany Colon  6011012799432041    Fri8382
99     Edward Carter   347435564751626    Fri5575
220    Ricky Johnson   213109508670736    Fri4607
97      Eric Herrera      580116092652    Fri9268
..               ...               ...        ...
144      Linda Jones  6542729219645658   Thur9002
203      Toni Brooks  3582289985920239   Thur7770
119   Melanie Jordan      676212062720   Thur8063
81    William Martin  4550549048402707   Thur8232
127  Jessica Simmons   213102478792200   Thur1512

[244 rows x 11 columns]
```

```
# Notiamo quindi che i giorni si ripetono: Ad esempio, i primi 5 sono
Fri. In questo caso conviene inserire un'ordinamento anche per
un'altra colonna.
# Ad esempio,supponiamo di voler ordinare il Df per day e per size.
Otterremo che:


tips = tips.sort_values(by=["day","size"])
tips
```

```
     total_bill   tip     sex smoker   day    time  size
price_per_person  \
222        8.58  1.92    Male    Yes   Fri   Lunch     1
8.58
92         5.75  1.00  Female    Yes   Fri  Dinner     2
2.88
101       15.38  3.00  Female    Yes   Fri  Dinner     2
7.69
99        12.46  1.50    Male     No   Fri  Dinner     2
6.23
220       12.16  2.20    Male    Yes   Fri   Lunch     2
6.08
..          ...   ...     ...    ...   ...     ...   ...
...
119       24.08  2.92  Female     No  Thur   Lunch     4
6.02
142       41.19  5.00    Male     No  Thur   Lunch     5
8.24
125       29.80  4.20  Female     No  Thur   Lunch     6
4.97
143       27.05  5.00  Female     No  Thur   Lunch     6
4.51
141       34.30  6.70    Male     No  Thur   Lunch     6
5.72

         Payer Name         CC Number Payment ID
222  Jason Lawrence  3505302934650403    Fri6624
92     Leah Ramirez  3508911676966392    Fri3780
101   Tiffany Colon  6011012799432041    Fri8382
99    Edward Carter   347435564751626    Fri5575
220   Ricky Johnson   213109508670736    Fri4607
..              ...              ...        ...
119  Melanie Jordan      676212062720   Thur8063
142    Eric Andrews  4356531761046453   Thur3621
125  Angela Sanchez      503857080488   Thur3948
143    Regina Jones     4311048695487   Thur6179
141  Steven Carlson  3526515703718508   Thur1025

[244 rows x 11 columns]
```

```
# CVD


# ESTRAZIONE INDICE DEI VALORI MASSIMO E MINIMO.
# CONSIDERIAMO LA VARIABILE TIP.
# lA DOMANDA QUINDI E': IN QUALE RIGA SI TROVANO IL VALORE MASSIMO E
IL VALORE MINIMO DELLA VARIABILE TIP?


# RIGA CONTENENTE IL VALORE MASSIMO


tips["tip"].idxmax()
```

```
170
```

```
# RIGA CONTENENTE IL VALORE MINIMO

tips["tip"].idxmin()
```

```
92
```

```
# Per estrarre tutta la riga corrispondente al valore massimo (lo
stesso varrà per il valore minimo)


tips.iloc[170]
```

```
total_bill                      31.71
tip                               4.5
sex                              Male
smoker                             No
day                               Sun
time                           Dinner
size                                4
price_per_person                 7.93
Payer Name             Michael Lawson
CC Number           3566285921227119
Payment ID                   Sun3719
Name: 167, dtype: object
```

```
# Possiamo quindi direttamente estrarre la riga corrispondente al
valore massimo della colonna tip:

tips.iloc[tips["tip"].idxmin()]
```

```
total_bill                    24.01
tip                             2.0
sex                            Male
smoker                          Yes
day                             Sat
time                         Dinner
size                              4
price_per_person                6.0
Payer Name          Michael Osborne
CC Number             4258682154026
Payment ID                  Sat7872
Name: 230, dtype: object
```

```
# CORRELAZIONE TRA VARIABILI. Si utilizza il metodo corr. Se non si
specifica alcun parametro,
# Pandas proverà a prendere in considerazione TUTTI i tipi di
variabili e, a meno che il Df non
# abbia solo variabili numerche, otterremo un errore.
# Per risolvere il problema, si insueriche numeric_only=True come
argomento della nostra funzione corr

tips.corr(numeric_only=True)
```

```
                 total_bill       tip      size  price_per_person  CC
Number
total_bill         1.000000  0.675734  0.598315          0.647554
0.104576
tip                0.675734  1.000000  0.489299          0.347405
0.110857
size               0.598315  0.489299  1.000000         -0.175359  -
0.030239
price_per_person   0.647554  0.347405 -0.175359          1.000000
0.135240
CC Number          0.104576  0.110857 -0.030239          0.135240
1.000000
```

```
# Come si legge la matrice di correlazione?
```

La matrice di correlazione in statistica è uno strumento che mostra il grado di relazione (correlazione) tra più variabili quantitative. È una tabella quadrata in cui:

Le righe e le colonne rappresentano le stesse variabili.

Ogni cella della matrice contiene il coefficiente di correlazione tra due variabili.

Il più usato è il coefficiente di Pearson, che misura la relazione lineare tra due variabili. I suoi valori vanno da:

+1 → correlazione perfettamente positiva (quando una aumenta, anche l'altra aumenta)

0 → nessuna correlazione lineare

–1 → correlazione perfettamente negativa (quando una aumenta, l'altra diminuisce)

Supponiamo di avere un dataset con queste 3 variabili quantitative:

Altezza Peso Età 170 65 25 180 75 30 … … …

La matrice di correlazione potrebbe essere:

Altezza Peso Età Altezza 1.00 0.85 0.30 Peso 0.85 1.00 0.40 Età 0.30 0.40 1.00

 Caratteristiche principali La diagonale è sempre 1, perché ogni variabile è perfettamente correlata con sé stessa.

La matrice è simmetrica: la correlazione tra A e B è uguale a quella tra B e A.

```
# value_counts(). L'abbiamo già utilizzato. Ci dice quali sono le
variabili categoriche e quante sono
# le loro occorrenze. ESEMPIO


tips["day"].value_counts()
```

```
day
Sat     87
Sun     76
Thur    62
Fri     19
Name: count, dtype: int64
```

```
# La colonna day presenta quindi 4 valori distinti. Il sabato compare
87 volte, La domenica 76, etc


# Se vogliamo sapere quali sono i valori distinti che una variabile
assume senza sapere il numero
# di occorrenze, si utilizza la funzione unique:


tips["day"].unique()
```

```
array(['Fri', 'Sat', 'Sun', 'Thur'], dtype=object)
```

```
# Per sapere, invece, QUANTI valori una certa variabile categorica
assume, si utilizza nunique:

tips["day"].nunique()
```

```
4
```

```
# Al posti di nunique possiamo contare i valori distinti assunti da
una variabile tramite
# la funzione len:

len(tips["day"].unique())
```

```
4
```

```
# Metodo replace


# Supponiamo di voler cambiare i nomi dei giorni della settimana. Ad
esempio, supponiamo di volerli
# Tradurre in italiano. Possiamo usare il metodo replace oppure il
metodo map. Il primo accetta due liste in input,
# il secondo utilizza un dizionario.

#- METODO REPLACE

tips["day"].unique()
```

```
array(['Fri', 'Sat', 'Sun', 'Thur'], dtype=object)
```

```
tips["day"].replace(["Fri","Sat","Sun","Thur"],
["Venerdì","Sabato","Domenica","Giovedì"])
```

```
222    Venerdì
92     Venerdì
101    Venerdì
99     Venerdì
220    Venerdì
       ...
119    Giovedì
142    Giovedì
125    Giovedì
143    Giovedì
141    Giovedì
Name: day, Length: 244, dtype: object
```

```
tips["day"].unique()
```

```
array(['Fri', 'Sat', 'Sun', 'Thur'], dtype=object)
```

```
# GRRRRRR i valori originari non sono stati cambiati!! Quindi:

tips["day"] = tips["day"].replace(["Fri","Sat","Sun","Thur"],
["Venerdì","Sabato","Domenica","Giovedì"])

tips["day"].unique()
```

```
array(['Venerdì', 'Sabato', 'Domenica', 'Giovedì'], dtype=object)
```

```
tips.head()
```

```
     total_bill   tip     sex smoker      day    time  size
price_per_person  \
222        8.58  1.92    Male    Yes  Venerdì   Lunch     1
8.58
92         5.75  1.00  Female    Yes  Venerdì  Dinner     2
2.88
101       15.38  3.00  Female    Yes  Venerdì  Dinner     2
7.69
99        12.46  1.50    Male     No  Venerdì  Dinner     2
6.23
220       12.16  2.20    Male    Yes  Venerdì   Lunch     2
6.08

         Payer Name         CC Number Payment ID
222  Jason Lawrence  3505302934650403    Fri6624
92     Leah Ramirez  3508911676966392    Fri3780
101   Tiffany Colon  6011012799432041    Fri8382
99    Edward Carter   347435564751626    Fri5575
220   Ricky Johnson   213109508670736    Fri4607
```

```
# PERFECT!!!


# METODO map

# Ora vogliamo vedere solo le prime 3 lettere dei giorni della
settimana in italiano, utilizzando il
# metodo map

tips["day"].unique()
```

```
array(['Venerdì', 'Sabato', 'Domenica', 'Giovedì'], dtype=object)
```

```
mymap =
{'Venerdì':'VEN','Sabato':'SAB','Domenica':'DOM','Giovedì':'GIO'}
```

```
tips["day"] = tips["day"].map(mymap)
```

```
tips["day"].unique()
```

```
array(['VEN', 'SAB', 'DOM', 'GIO'], dtype=object)
```

```
# CVD
```

```
# In sintesi, si utilizza il metodo replace quando abbiamo a che fare
con un solo valore
# da modificare, mentre si utilizza map quando abbiamo a che fare con
più di un valore
```

```
# OPERATORE BETWEEN
```

```
# Considera solo le variabili che si trovano in un determinato range.
# Prendiamo in considerazione la variabile price_per_person.
Supponiamo di voler estrarre solo i valori
# compresi tra 5 e 8. Possiamo ragionare sia tramite l'uso dei filtri
sia tramite l'operatore between
```

```
price_per_person = tips["price_per_person"]
```

```
# METODO CON I FILTRI
```

```
mask = (price_per_person >= 5) & (price_per_person <= 8)
tips[mask]
```

```
     total_bill   tip     sex smoker  day    time  size
price_per_person  \
101       15.38  3.00  Female    Yes  VEN  Dinner     2
7.69
99        12.46  1.50    Male     No  VEN  Dinner     2
6.23
220       12.16  2.20    Male    Yes  VEN   Lunch     2
6.08
97        12.03  1.50    Male    Yes  VEN  Dinner     2
6.02
```

```
100        11.35  2.50  Female    Yes  VEN  Dinner     2
5.68
..           ...   ...     ...    ...  ...     ...   ...
...
205        16.47  3.23  Female    Yes  GIO   Lunch     3
5.49
204        20.53  4.00    Male    Yes  GIO   Lunch     4
5.13
77         27.20  4.00    Male     No  GIO   Lunch     4
6.80
119        24.08  2.92  Female     No  GIO   Lunch     4
6.02
141        34.30  6.70    Male     No  GIO   Lunch     6
5.72

          Payer Name         CC Number Payment ID
101    Tiffany Colon  6011012799432041    Fri8382
99     Edward Carter   347435564751626    Fri5575
220    Ricky Johnson   213109508670736    Fri4607
97      Eric Herrera      580116092652    Fri9268
100       Lori Lynch    38558279384492    Fri4106
..               ...               ...        ...
205      Carly Reyes     4787787236486   Thur8084
204        Scott Kim  3570611756827620   Thur2160
77        John Davis    30344778738589   Thur4924
119  Melanie Jordan      676212062720   Thur8063
141  Steven Carlson  3526515703718508   Thur1025

[120 rows x 11 columns]


# METODO CON BETWEEN


mask = price_per_person.between(5,8)

tips[mask]

     total_bill   tip     sex smoker  day    time  size
price_per_person  \
101        15.38  3.00  Female    Yes  VEN  Dinner     2
7.69
99         12.46  1.50    Male     No  VEN  Dinner     2
6.23
220        12.16  2.20    Male    Yes  VEN   Lunch     2
6.08
97         12.03  1.50    Male    Yes  VEN  Dinner     2
6.02
100        11.35  2.50  Female    Yes  VEN  Dinner     2
```

```
5.68
..          ...   ...     ...    ...  ...     ...   ...
...
205       16.47  3.23  Female    Yes  GIO   Lunch     3
5.49
204       20.53  4.00    Male    Yes  GIO   Lunch     4
5.13
77        27.20  4.00    Male     No  GIO   Lunch     4
6.80
119       24.08  2.92  Female     No  GIO   Lunch     4
6.02
141       34.30  6.70    Male     No  GIO   Lunch     6
5.72

         Payer Name         CC Number Payment ID
101   Tiffany Colon  6011012799432041    Fri8382
99    Edward Carter   347435564751626    Fri5575
220   Ricky Johnson   213109508670736    Fri4607
97     Eric Herrera      580116092652    Fri9268
100      Lori Lynch    38558279384492    Fri4106
..               ...               ...        ...
205     Carly Reyes     4787787236486   Thur8084
204       Scott Kim  3570611756827620   Thur2160
77       John Davis    30344778738589   Thur4924
119  Melanie Jordan      676212062720   Thur8063
141  Steven Carlson  3526515703718508   Thur1025

[120 rows x 11 columns]


# nlargest ed nsmallest


# nlarghest => sorta il dataframe in base alla colonna desiderata in
senso decrescente.
# Il primo argomento corrisponde al numero di colonne che si vuole
visualizzare,
# mentre il secondo argomento è il nome della colonna


tips.nlargest(6,'price_per_person')

     total_bill    tip   sex smoker  day    time  size
price_per_person  \
184       40.55   3.00  Male    Yes  DOM  Dinner     2
20.27
179       34.63   3.55  Male    Yes  DOM  Dinner     2
17.32
```

```
170       50.81  10.00  Male    Yes  SAB  Dinner     3
16.94
175       32.90   3.11  Male    Yes  DOM  Dinner     2
16.45
237       32.83   1.17  Male    Yes  SAB  Dinner     2
16.42
83        32.68   5.00  Male    Yes  GIO   Lunch     2
16.34

          Payer Name         CC Number Payment ID
184      Stephen Cox  3547798222044029    Sun5140
179     Brian Bailey   346656312114848    Sun9851
170    Gregory Clark  5473850968388236    Sat1954
175  Nathan Reynolds   370307040837149    Sun5109
237     Thomas Brown  4284722681265508    Sat2929
83     Daniel Murphy  5356177501009133   Thur8801


# nsmallest => sorta il dataframe in base alla colonna desiderata in
senso crescente.
# Il primo argomento corrisponde al numero di colonne che si vuole
visualizzare,
# mentre il secondo argomento è il nome della colonna


tips.nsmallest(6,'price_per_person')

     total_bill   tip     sex smoker  day    time  size
price_per_person  \
92         5.75  1.00  Female    Yes  VEN  Dinner     2
2.88
67         3.07  1.00  Female    Yes  SAB  Dinner     1
3.07
16        10.33  1.67  Female     No  DOM  Dinner     3
3.44
1         10.34  1.66    Male     No  DOM  Dinner     3
3.45
172        7.25  5.15    Male    Yes  DOM  Dinner     2
3.62
149        7.51  2.00    Male     No  GIO   Lunch     2
3.76

           Payer Name         CC Number Payment ID
92       Leah Ramirez  3508911676966392    Fri3780
67      Tiffany Brock  4359488526995267    Sat3455
16   Elizabeth Foster  4240025044626033    Sun9715
1      Douglas Tucker  4478071379779230    Sun4608
172       Larry White    30432617123103    Sun9209
149    Daniel Robbins  4823139288341889   Thur6321
```

```
# SAMPLE


# Possiamo estrarre n righe casuali dal nostro Df:


tips.sample(2)
```

```
     total_bill  tip     sex smoker  day    time  size
price_per_person  \
6          8.77  2.0    Male     No  DOM  Dinner     2
4.38
136       10.33  2.0  Female     No  GIO   Lunch     2
5.16

              Payer Name         CC Number Payment ID
6    Kristopher Johnson  2223727524230344    Sun5985
136         Donna Kelly   180048553626376   Thur1393
```

```
# Ha estratto due righe a cacchio. Si noti che l'ordine non viene
sempre rispettato.
```