

Python - Analiza danych z modulem PANDAS

www.udemy.com (<http://www.udemy.com>) (R)

LAB - S05-L006 - pivot_table

1. Zaimportuj moduł pandas i numpy nadaj im standardowe aliasy. Zaimportuj też datetime, timedelta i time, możesz skorzystać z poniższych poleceń:

```
from datetime import datetime
from datetime import timedelta
import time
```

1. Do zmiennej **df** zaimportuj plik **marathon_results_2016.csv**. Jako indeks wybierz kolumnę **Bip**. Zaimportuj kolumny: **'Bib','40K','Half','Pace','Age','M/F','Country','State','City'**. Znaczenie kolumn to kolejno: numer zawodnika, czas na 40 km, czas na 20 km, tempo liczone w minutach na milę, wiek, płeć, kraj, stan i miasto. Korzystając z poniższego kodu skonwertuj zawartość kolumn do typu **timedelta**:

```
df['40K'] = df['40K'].apply(pd.to_timedelta)
df['Half'] = df['Half'].apply(pd.to_timedelta)

df['TotalSeconds'] = df['40K'].apply(lambda x: timedelta.total_seconds(x)
)
df['HalfSeconds'] = df['Half'].apply(lambda x: timedelta.total_seconds(x)
)
```

3. Wyświetl tabelę przestawną, która pozwoli porównać średni czas biegu na dystansie 40 km dla zawodników w różnym wieku z podziałem na mężczyzn i kobiety: w wierszach ma być wiek zawodników (kolumna **Age**), w kolumnach informacja o płci (kolumna **M/F**), w komórkach średnia z czasu **TotalSeconds**
4. Do poprzedniej tabeli dodaj również dane o średnim czasie **HalfSeconds**
5. Zmień kolejność poziomów dla kolumn. Najpierw ma być dokonany podział na mężczyzn i kobiety, a dopiero potem rozróżnienie średnich czasów na dystansie 20 i 40 km
6. Zmień sortowanie kolumn w tabeli z poprzedniego przykładu tak, aby nagłówki dla kobiet i mężczyzn pojawił się tylko jeden raz.
7. A właściwie to ilu było zawodników? Zbuduj tabelę przestawną, która zaprezentuje w wierszach wiek uczestników, a w kolumnie ilość uczestników z podziałem na płeć. Do liczenia zawodników może być wykorzystane dowolne z dostępnych pól.
8. Zmień poprzednie polecenie tak, aby wartości były wyznaczane bez podziału na kobiety i mężczyzn
9. Zbuduj tabelę, która policzy ilość uczestników z różnych krajów. Kraje mają się znaleźć w wierszach.
10. Zmień poprzednie polecenie tak, aby wiersze były posortowane wg ilości zawodników z poszczególnych krajów malejąco.
11. Zbuduj tabelę przestawną, która będzie prezentować średni czas biegu uzyskany przez reprezentantów poszczególnych miast. Tabela ma posiadać multiindeks w oparciu o kolumny **County, State, City**
12. Posortuj dane z poprzedniego polecenia wg czasu malejąco. W ten sposób wyznaczasz miasto z najlepszymi średnimi wynikami.

Dane pochodzą z <https://github.com/llimllib/bostonmarathon> (<https://github.com/llimllib/bostonmarathon>)
<https://www.kaggle.com/rojour/boston-marathon-2016-finishers-analysis/data> (<https://www.kaggle.com/rojour/boston-marathon-2016-finishers-analysis/data>)

Rozwiązania:

Poniżej znajdują się propozycje rozwiązań zadań. Prawdopodobnie istnieje wiele dobrych rozwiązań, dlatego jeżeli rozwiązujesz zadania samodzielnie, to najprawdopodobniej zrobisz to inaczej, może nawet lepiej :) Możesz pochwalić się swoimi rozwiązaniami w sekcji Q&A

```
In [1]: import pandas as pd
import numpy as np
from datetime import datetime
from datetime import timedelta
import time
```

```
In [2]: df = pd.read_csv('./marathon_results_2016.csv', index_col='Bib',
                        usecols=['Bib', '40K', 'Half', 'Pace', 'Age', 'M/F', 'Country', 'State', 'City'])

df['40K'] = df['40K'].apply(pd.to_timedelta)
df['Half'] = df['Half'].apply(pd.to_timedelta)

df['TotalSeconds'] = df['40K'].apply(lambda x: timedelta.total_seconds(x))
df['HalfSeconds'] = df['Half'].apply(lambda x: timedelta.total_seconds(x))

df.head()
```

```
Out[2]:
```

	Age	M/F	City	State	Country	Half	40K	Pace	TotalSeconds	HalfSeconds
Bib										
5	21	M	Addis Ababa	NaN	ETH	01:06:45	02:05:59	0:05:04	7559.0	4005.0
1	26	M	Ambo	NaN	ETH	01:06:46	02:05:59	0:05:06	7559.0	4006.0
6	31	M	Addis Ababa	NaN	ETH	01:06:44	02:06:47	0:05:07	7607.0	4004.0
11	33	M	Kitale	NaN	KEN	01:06:46	02:06:47	0:05:07	7607.0	4006.0
14	23	M	Eldoret	NaN	KEN	01:06:46	02:08:11	0:05:11	7691.0	4006.0

```
In [3]: df.pivot_table(index="Age", columns="M/F", values="TotalSeconds").head()
```

```
Out[3]:
```

	M/F	F	M
Age			
18	16050.666667	14554.615385	
19	15351.958333	12156.529412	
20	13835.931818	12111.100000	
21	14681.027027	12408.360465	
22	14366.421053	11872.666667	

```
In [4]: df.pivot_table(index="Age", columns="M/F", values=["HalfSeconds", "TotalSeconds"]).head()
```

```
Out[4]:
```

	HalfSeconds		TotalSeconds	
	M/F	F	M	F
Age				
18		7999.888889	7078.538462	16050.666667
19		7473.666667	5817.764706	15351.958333
20		6871.045455	5716.800000	13835.931818
21		7148.554054	5929.627907	14681.027027
22		7091.097744	5751.000000	14366.421053

```
In [5]: df.pivot_table(index="Age", columns="M/F", values=["HalfSeconds", "TotalSeconds"]
        ).swaplevel(axis=1).head()
```

```
Out[5]:
```

	M/F	F	M	F	M
		HalfSeconds	HalfSeconds	TotalSeconds	TotalSeconds
Age					
18		7999.888889	7078.538462	16050.666667	14554.615385
19		7473.666667	5817.764706	15351.958333	12156.529412
20		6871.045455	5716.800000	13835.931818	12111.100000
21		7148.554054	5929.627907	14681.027027	12408.360465
22		7091.097744	5751.000000	14366.421053	11872.666667

```
In [6]: df.pivot_table(index="Age", columns="M/F", values=["HalfSeconds", "TotalSeconds"]
        ).swaplevel(axis=1).sort_index(axis=1).head()
```

```
Out[6]:
```

	M/F	F	M	
		HalfSeconds	TotalSeconds	HalfSeconds
Age				TotalSeconds
18		7999.888889	16050.666667	7078.538462
19		7473.666667	15351.958333	5817.764706
20		6871.045455	13835.931818	5716.800000
21		7148.554054	14681.027027	5929.627907
22		7091.097744	14366.421053	5751.000000

```
In [7]: df.pivot_table(index="Age", columns="M/F", aggfunc='count', values='City').head()
```

```
Out[7]:
```

	M/F	F	M
Age			
18		9.0	13.0
19		24.0	17.0
20		44.0	40.0
21		74.0	86.0
22		133.0	93.0

```
In [8]: df.pivot_table(index="Country", aggfunc='count', values='City').head()
```

```
Out[8]:
```

	City
Country	
ALB	1
AND	1
ARG	26
AUS	152
AUT	27

```
In [9]: df.pivot_table(index="Country", aggfunc='count', values='City'
                        ).sort_values('City', ascending=False).head()
```

```
Out[9]:
```

	City
Country	
USA	21649
CAN	2134
GBR	366
MEX	252
JPN	185

```
In [10]: df.pivot_table(index=["Country", "State", "City"], aggfunc='mean',
                        values='TotalSeconds').head()
```

```
Out[10]:
```

			TotalSeconds
Country	State	City	
CAN	AB	Airdrie	11211.000000
		Ardrossan	12644.000000
		Calgary	12546.186813
		Cochrane	12879.500000
		Devon	14414.500000

```
In [11]: df.pivot_table(index=["Country", "State", "City"],
                        aggfunc='mean', values='TotalSeconds'
                        ).sort_values('TotalSeconds', ascending=False).head()
```

Out[11]:

TotalSeconds			
Country	State	City	
USA	WI	Reedsburg	28461.0
		Justin	23816.0
	MS	Olive Branch	21878.0
	FL	Estero	21468.0
	TX	Fresno	20919.0

In []: