Python - Analiza danych z modułem PANDAS

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

LAB - S06-L001 - wprowadzenie do grupowania

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
```

2. Do wykonania zadań będziemy korzystać z danych dotyczących maratonów. Uruchom poniższy kod, który przygotuje zmienną df o odpowiedniej strukturze:

```
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()
```

3. Wyświetl informacje o pobranych danych korzystając z:

   - info()
   - describe()
   - value_counts()
   - unique()
   - nunique()

4. Do nowej zmiennej **cities** zapisz unikalne nazwy miast (kolumna **City**) z obiektu data frame
5. Utwórz pusty słownik o nazwie **groups**
6. Podobnie jak w materiale video, przygotuj pętlę, która dla każdego elementu z listy **cities** pobierze z oryginalnego obiektu **df** wiersze pasujące do w danej chwili przetwarzanego miasta. Do zmiennej **groups** dodaj nową parę klucz/wartość, gdzie kluczem jest nazwa kraju, a wartością wszystkie te wiersze z obiektu **df**, które opisują wyniki maratonów w tych krajach.
7. Poleceniem describe porównaj wyniki uzyskiwane w kilku wybranych miastach np. "San Franciso" i "Addis Ababa"

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]:

Bib	Age	M/F	City	State	Country	Half	40K	Pace	TotalSeconds	HalfSeconds
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.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 26630 entries, 5 to 28487
Data columns (total 10 columns):
Age             26630 non-null int64
M/F             26630 non-null object
City            26629 non-null object
State           23783 non-null object
Country         26630 non-null object
Half            26630 non-null timedelta64[ns]
40K             26630 non-null timedelta64[ns]
Pace            26630 non-null object
TotalSeconds    26630 non-null float64
HalfSeconds     26630 non-null float64
dtypes: float64(2), int64(1), object(5), timedelta64[ns](2)
memory usage: 2.2+ MB
```

In [4]:
```
df.describe()
```

Out[4]:

	Age	Half	40K	TotalSeconds	HalfSeconds
count	26630.000000	26630	26630	26630.000000	26630.000000
mean	42.514044	0 days 01:49:25.497070	0 days 03:41:49.984829	13309.984829	6565.497071
std	11.347955	0 days 00:18:06.943013	0 days 00:39:13.302840	2353.302841	1086.943013
min	18.000000	0 days 00:00:00	0 days 00:00:00	0.000000	0.000000
25%	34.000000	0 days 01:36:51	0 days 03:14:50.250000	11690.250000	5811.000000
50%	43.000000	0 days 01:47:00	0 days 03:35:13	12913.000000	6420.000000
75%	51.000000	0 days 01:58:32	0 days 04:02:20	14540.000000	7112.000000
max	83.000000	0 days 04:47:17	0 days 07:59:12	28752.000000	17237.000000

In [5]:
```
df["City"].value_counts().head()
```

Out[5]:
```
Boston           722
New York         451
Chicago          261
Toronto          223
San Francisco    197
Name: City, dtype: int64
```

In [6]:
```
df["City"].unique()
```

Out[6]:
```
array(['Addis Ababa', 'Ambo', 'Kitale', ..., 'Justin', 'Rossmoor',
       'Reedsburg'], dtype=object)
```

In [7]:
```
df["City"].nunique()
```

Out[7]:
```
5832
```

In [8]:
```
cities = df["City"].unique()
```

In [9]:
```
groups = {}
```

In [10]:
```
for city in cities:
    tmp_df = df.loc[ df["City"] == city]
    groups[city] = tmp_df
```

In [11]:
```
groups["San Francisco"].describe()
```

Out[11]:

	Age	Half	40K	TotalSeconds	HalfSeconds
count	197.000000	197	197	197.000000	197.000000
mean	36.309645	0 days 01:42:24.390862	0 days 03:25:35.335025	12335.335025	6144.390863
std	10.600373	0 days 00:15:31.836366	0 days 00:34:13.037100	2053.037101	931.836367
min	22.000000	0 days 01:14:21	0 days 02:29:21	8961.000000	4461.000000
25%	28.000000	0 days 01:30:11	0 days 03:02:06	10926.000000	5411.000000
50%	34.000000	0 days 01:41:50	0 days 03:21:13	12073.000000	6110.000000
75%	42.000000	0 days 01:48:48	0 days 03:39:35	13175.000000	6528.000000
max	80.000000	0 days 02:45:21	0 days 05:39:13	20353.000000	9921.000000

In [12]:
```
groups["Addis Ababa"].describe()
```

Out[12]:

	Age	Half	40K	TotalSeconds	HalfSeconds
count	5.00000	5	5	5.000000	5.000000
mean	25.00000	0 days 01:08:30.200000	0 days 02:13:47.600000	8027.600000	4110.200000
std	4.84768	0 days 00:03:55.793765	0 days 00:10:07.503744	607.503745	235.793766
min	20.00000	0 days 01:06:44	0 days 02:05:59	7559.000000	4004.000000
25%	21.00000	0 days 01:06:45	0 days 02:06:47	7607.000000	4005.000000
50%	24.00000	0 days 01:06:45	0 days 02:10:57	7857.000000	4005.000000
75%	29.00000	0 days 01:06:45	0 days 02:14:23	8063.000000	4005.000000
max	31.00000	0 days 01:15:32	0 days 02:30:52	9052.000000	4532.000000

In [ ]:
```

```