Porownanie_do_Pythona

May 17, 2022

1 PodstawyBigData2122LS

Autorzy:

- Bartosz Durys 229869
- Bartosz Niciak 229969

[1]:
```
import pandas as pd
import time
```

[2]:
```
salaries_df = pd.read_csv('Salaries.csv', header=0)
salaries_df
```

[2]:
```
       yearID teamID lgID   playerID   salary
0        1985    BAL   AL  murraed02  1472819
1        1985    BAL   AL   lynnfr01  1090000
2        1985    BAL   AL  ripkeca01   800000
3        1985    BAL   AL   lacyle01   725000
4        1985    BAL   AL  flanami01   641667
…         …     …   …         …        …
23951    2013    WAS   NL  matthry01   504500
23952    2013    WAS   NL  lombast02   501250
23953    2013    WAS   NL  ramoswi01   501250
23954    2013    WAS   NL  rodrihe03   501000
23955    2013    WAS   NL  moorety01   493000

[23956 rows x 5 columns]
```

[3]:
```
batting_df = pd.read_csv('Batting.csv', header=0)
batting_df
```

[3]:
```
        playerID  yearID  stint teamID lgID    G  G_batting     AB     R  \
0      aardsda01    2004      1    SFN   NL   11       11.0    0.0   0.0
1      aardsda01    2006      1    CHN   NL   45       43.0    2.0   0.0
2      aardsda01    2007      1    CHA   AL   25        2.0    0.0   0.0
3      aardsda01    2008      1    BOS   AL   47        5.0    1.0   0.0
4      aardsda01    2009      1    SEA   AL   73        3.0    0.0   0.0
…          …      …     …    …   …  …  …         …     …   …
```

```
97884  zimmejo02    2013      1    WAS   NL   32       32.0   65.0   4.0
97885  zimmery01    2013      1    WAS   NL  147      147.0  568.0  84.0
97886   zitoba01    2013      1    SFN   NL   30       30.0   34.0   3.0
97887  zobribe01    2013      1    TBA   AL  157      157.0  612.0  77.0
97888  zuninmi01    2013      1    SEA   AL   52       52.0  173.0  22.0

            H  …    SB   CS    BB     SO  IBB  HBP   SH   SF  GIDP  G_old
0         0.0  …   0.0  0.0   0.0    0.0  0.0  0.0  0.0  0.0   0.0   11.0
1         0.0  …   0.0  0.0   0.0    0.0  0.0  0.0  1.0  0.0   0.0   45.0
2         0.0  …   0.0  0.0   0.0    0.0  0.0  0.0  0.0  0.0   0.0    2.0
3         0.0  …   0.0  0.0   0.0    1.0  0.0  0.0  0.0  0.0   0.0    5.0
4         0.0  …   0.0  0.0   0.0    0.0  0.0  0.0  0.0  0.0   0.0    NaN
…       …  …    …   …    …     …    …  …   …  …   …    …     …
97884     8.0  …   0.0  0.0   1.0   20.0  0.0  0.0  6.0  1.0   0.0    NaN
97885   156.0  …   6.0  0.0  60.0  133.0  2.0  2.0  0.0  3.0  16.0    NaN
97886     5.0  …   0.0  0.0   0.0    8.0  0.0  0.0  9.0  0.0   1.0    NaN
97887   168.0  …  11.0  3.0  72.0   91.0  4.0  7.0  1.0  6.0  18.0    NaN
97888    37.0  …   1.0  0.0  16.0   49.0  0.0  3.0  0.0  1.0   5.0    NaN

[97889 rows x 24 columns]
```

2 1. identyfikatory wszystkich drużyn (teamID), w których w roku 1985 każdy gracz zarabiał powyżej 100000.

[4]:
```
start_time = time.time()

salaries_min_1985_df = salaries_df[salaries_df['yearID']==1985].
 ↪groupby('teamID').min()
teamID_list = salaries_min_1985_df[salaries_min_1985_df['salary']>100000].index.
 ↪tolist()

end_time = time.time()

print('Rozwiązanie:', teamID_list)
print(f'Operacja zajęła {end_time-start_time} sekund')
```

```
Rozwiązanie: ['ATL', 'BOS', 'CHN', 'CLE', 'SDN']
Operacja zajęła 0.009563446044921875 sekund
```

3 2. identyfikatory wszystkich drużyn (teamID) wraz ze średnim wynagrodzeniem w tych drużynach w 1998.

[5]:
```
start_time = time.time()

salaries_avg_1998_df = salaries_df[salaries_df['yearID']==1998].
 ↪groupby('teamID').mean()['salary']

end_time = time.time()

print('Rozwiązanie:', salaries_avg_1998_df)
print(f'Operacja zajęła {end_time-start_time} sekund')
```

```
Rozwiązanie: teamID
ANA    1.214147e+06
ARI    8.985278e+05
ATL    1.912062e+06
BAL    2.411854e+06
BOS    1.719909e+06
CHA    1.161667e+06
CHN    1.639935e+06
CIN    6.766176e+05
CLE    1.842429e+06
COL    1.682822e+06
DET    6.875714e+05
FLO    1.033067e+06
HOU    1.324188e+06
KCA    9.451923e+05
LAN    1.435882e+06
MIL    9.974972e+05
MIN    9.630172e+05
MON    3.224697e+05
NYA    2.087715e+06
NYN    1.578121e+06
OAK    6.086571e+05
PHI    1.037071e+06
PIT    4.304286e+05
SDN    1.562050e+06
SEA    1.423343e+06
SFN    1.520208e+06
SLN    1.562072e+06
TBA    8.266667e+05
TEX    1.885736e+06
TOR    1.605500e+06
Name: salary, dtype: float64
Operacja zajęła 0.007524251937866211 sekund
```

4 3. dla każdej ligii (lgID) liczbę drużyn w 1999.

[6]:
```
start_time = time.time()

teams = salaries_df[salaries_df['yearID']==1999].groupby('lgID').
 ↪nunique()['teamID']

end_time = time.time()

print('Rozwiązanie:', teams)
print(f'Operacja zajęła {end_time-start_time} sekund')
```

```
Rozwiązanie: lgID
AL    14
NL    16
Name: teamID, dtype: int64
Operacja zajęła 0.007472991943359375 sekund
```

5 4. top 10 graczy z największą liczbą uderzeń (H - hits) w 1988. Wypisz playerID i liczbę tych uderzeń.

[7]:
```
start_time = time.time()

top_10_player_hits = batting_df[batting_df['yearID']==1988].nlargest(10,␣
 ↪'H')[['playerID', 'H']]

end_time = time.time()

print('Rozwiązanie:\n', top_10_player_hits)
print(f'Operacja zajęła {end_time-start_time} sekund')
```

```
Rozwiązanie:
         playerID      H
70350  puckeki01  234.0
7667   boggswa01  214.0
32915  greenmi01  192.0
59948  molitpa01  190.0
96125  yountro01  190.0
12957  cansejo01  187.0
26427  fernato01  186.0
28358  francju01  186.0
55050  mattido01  186.0
29365  galaran01  184.0
Operacja zajęła 0.01537179946899414 sekund
```

6 5. graczy, którzy rozgrywali najwięcej gier w 1980 (G_batting - kolumna nr 7). Wypisz playerID i liczbę tych gier.

[8]:
```
start_time = time.time()

top_10_player_games = batting_df[batting_df['yearID']==1980].nlargest(10,␣
↪'G_batting')[['playerID', 'G_batting']]

end_time = time.time()

print('Rozwiązanie:\n', top_10_player_games)
print(f'Operacja zajęła {end_time-start_time} sekund')
```

```
Rozwiązanie:
         playerID  G_batting
30152  garvest01      163.0
64892  oliveal01      163.0
18624  cromawa01      162.0
46885  knighra01      162.0
60726  morenom01      162.0
61002  morriji01      162.0
75138   rosepe01      162.0
86480  thomago01      162.0
94390  winfida01      162.0
94305  wilsowi02      161.0
Operacja zajęła 0.005001068115234375 sekund
```

7 6. graczy, którzy w drużynie ML1 wykonali najwięcej biegów (runs - R). Wypisz playerID i liczbę R.

[9]:
```
start_time = time.time()

top_10_player_runs = batting_df[batting_df['teamID']=='ML1'].nlargest(10,␣
↪'R')[['playerID', 'R']]

end_time = time.time()

print('Rozwiązanie:\n', top_10_player_runs)
print(f'Operacja zajęła {end_time-start_time} sekund')
```

```
Rozwiązanie:
         playerID      R
15     aaronha01  127.0
16     aaronha01  121.0
10     aaronha01  118.0
54846  matheed01  118.0
```

```
12      aaronha01  116.0
14      aaronha01  115.0
10753   brutobi01  112.0
54840   matheed01  110.0
11      aaronha01  109.0
18      aaronha01  109.0
Operacja zajęła 0.011909008026123047 sekund
```

8 7. graczy, którzy zarobili w 2001 powyżej 500000 i mają więcej niż 50 biegów do bazy (homeruns - HR).

[10]:
```
start_time = time.time()

players_salary = set(salaries_df[(salaries_df['yearID']==2001) &␣
 ↪(salaries_df['salary']>500000)]['playerID'].unique())
players_batting = set(batting_df[batting_df['HR']>50]['playerID'].unique())
selected = players_salary & players_batting

end_time = time.time()

print('Rozwiązanie:\n', selected)
print(f'Operacja zajęła {end_time-start_time} sekund')
```

```
Rozwiązanie:
 {'thomeji01', 'bondsba01', 'gonzalu01', 'rodrial01', 'sosasa01', 'mcgwima01',
'griffke02', 'jonesan01'}
Operacja zajęła 0.0030007362365722656 sekund
```

9 Wnioski

- Kod napisany w języku Python wykonywany jest znacznie szybciej niż polecenia w PigLatin.
- PigLatin jest kompilowany do zadań MapReduce.
- W PigLatin średniki na końcu poleceń są obowiązkowe.
- Dane w Pythonie zostały umieszczone w pamięci operacyjnej, w PigLatin dane były zczytywane z pamięci masowej.