Praca Domowa 4

by Lizaveta Sauchanka

Spis treści

Rozdział I

Wprowadzenie

Praca Domowa 4 z przedmiotu PDU polegała na rozwiązaniu dwóch poleceń SQL przy uzyciu wywołan funkcji i metod z pakietu ***pandas***. Kazdemu z 2 polecen SQL odpowiadają dwa równowazne sposoby ich implementacji:

- wywołanie *pandas.read_sql_query("""zapytanie SQL""")*;
- wywołanie ciagu „zwykłych” metod i funkcji z pakietu *pandas*.

Pracowałam na uproszczonym zrzucie zanonimizowanych danych z serwisu https://travel.stackexchange.com/, a mianowicie na trzech zbiorach danych (ramkach danych) **Tags**, **Posts**, **Comments**.

Dodatkowo w tym raporcie będę sprawdzała równowaznosci wyników (Rozdział III).

Rozdział II

Pzeprowadzenie pracy z bazą danych

Aby zapewnić jakościowy kod i prawidłowe wyświetlanie wyników musimy podać następujące wstawki kodu:

In [2]:

```
# importujemy potrzepne biblioteki
import numpy as np
import pandas as pd
import os, os.path
import sqlite3
import tempfile

pd.set_option("display.notebook_repr_html", False)
```

In [3]:

```
# ładujemy potrzepne ramki danych
Tags = pd.read_csv("Tags.csv.gz",
                   compression = "gzip",
                   comment = "#")
Comments = pd.read_csv("Comments.csv.gz",
                   compression = "gzip",
                   comment = "#")
Posts = pd.read_csv("Posts.csv.gz",
                   compression = "gzip",
                   comment = "#")
```

In [4]:

```
# sciezka dostepu do bazy danych
baza = os.path.join(tempfile.mkdtemp(), "PD4.db")

if os.path.isfile(baza):
    os.remove(baza)

# połaczenie do bazy danych
conn = sqlite3.connect(baza)

# importujemy ramke danych do bazy danych
Tags.to_sql("Tags", conn)
Comments.to_sql("Comments", conn)
Posts.to_sql("Posts", conn)
```

Teraz możemy rozwiązywać zadania!

Rozdział II

Interpretacja zapytań oraz sprawzenie wyników równowaznosci

Zadanie 1

Przejdziemy do treści Zadania 1. Mamy zaimplementować następujące polecenie SQL:

```
SELECT TagName, Count
FROM Tags
ORDER BY Count DESC
LIMIT 10
```

Innymi słowy, mamy wypisać kolumny *TagName* oraz *Count* z ramki danych **Tags**, posortować malejąco względem kolumny *Count* i wypisać pierwsze 10 rzędów.

Teraz zobaczyć jak ma wyglądać ta ramka danych:

In [5]:

```
df_sql_1 = pd.read_sql_query("""
SELECT TagName, Count
FROM Tags
ORDER BY Count DESC
LIMIT 10
""", conn)

df_sql_1
```

Out[5]:

	TagName	Count
0	visas	5271
1	usa	2858
2	air-travel	2830
3	uk	2114
4	schengen	2094
5	customs-and-immigration	1798
6	transit	1204
7	trains	1031
8	passports	954
9	indian-citizens	916

Teraz przejdziemy do implementacji Zadania 1 w funkcjach z pakietu ***pandas***:

In [15]:

```
# wybieramy kolumny TagName i Count, sortujemy wierszy malejąco odnośnie kolumny Count,
# ustawiamy kolejność wierszy od 0 i wypisujemy pierwsze 10 wierszy
df_pd_1 = Tags.loc[:, ["TagName", "Count"]].sort_values("Count", ascending = False)\
                        .reset_index(drop = True)\
                        .head(10)

df_pd_1
```

Out[15]:

	TagName	Count
0	visas	5271
1	usa	2858
2	air-travel	2830
3	uk	2114
4	schengen	2094
5	customs-and-immigration	1798
6	transit	1204
7	trains	1031
8	passports	954
9	indian-citizens	916

Sprawdzimy teraz czy ramka danych **df_sql_1** jest identyczna ramce danych **df_pd_1** za pomocą funkcji *equals()*:

In [11]:

```
df_sql_1.equals(df_pd_1)
```

Out[11]:

```
True
```

Czyli **df_sql_1** oraz **df_pd_1** są identyczne.

Zadanie 2

Zobaczmy treść Zadania 2:

```
SELECT
    Posts.Title, Posts.CommentCount,
    CmtTotScr.CommentsTotalScore,
    Posts.ViewCount
FROM (
      SELECT
         PostID,
         UserID,
         SUM(Score) AS CommentsTotalScore
      FROM Comments
      GROUP BY PostID, UserID
) AS CmtTotScr
JOIN Posts ON Posts.ID=CmtTotScr.PostID
WHERE Posts.PostTypeId=1
ORDER BY CmtTotScr.CommentsTotalScore DESC
LIMIT 10
```

Zgodnie z treścią zadania, mamy najpierw stworzyć pomocniczą ramkę danych **CmtTotScr**. Następnie wybrane kolumny *Title*, *CommentCount* i *ViewCount* z **Posts** połączyć z wybranymi kolumnami *CommentsTotalScore* z ramki danych **CmtTotScr** (kluczem ramki **Posts** jest kolumna *Id*, natomiast kluczem ramki **CmtTotScr** jest kolumna *PostId*), posortować tak, aby w wynikowej ramce danych wartości kolumny *PostTypeId* z **Posts** były równe 1, posortować malejąco względem kolumny *CommentsTotalScore* i wypisać pierwsze 10 rzędów.

Tworzymy pomocniczą ramkę danych w następujący sposób: wybieramy kolumny *PostID* i *UserID* z ramki *Comments* oraz sumujemy i grupujemy wartości z *Score* z **Comments** odnośnie kolumn *PostID* i *UserID*.

Więc zobaczmy tą ramkę danych:

In [12]:

```
df_sql_2 = pd.read_sql_query("""
SELECT
     Posts.Title, Posts.CommentCount,
     CmtTotScr.CommentsTotalScore,
     Posts.ViewCount
FROM (
      SELECT
         PostID,
         UserID,
         SUM(Score) AS CommentsTotalScore
      FROM Comments
      GROUP BY PostID, UserID
) AS CmtTotScr
JOIN Posts ON Posts.ID=CmtTotScr.PostID
WHERE Posts.PostTypeId=1
ORDER BY CmtTotScr.CommentsTotalScore DESC
LIMIT 10
""", conn)

df_sql_2
```

Out[12]:

	Title	CommentCount	CommentsTotalScore	ViewCount
0	Boss is asking for passport, but it has a stam...	24	207	54982.0
1	Why don't airlines have backup planes just in ...	26	172	14516.0
2	OK we're all adults here, so really, how on ea...	27	155	73808.0
3	How to cross a road by foot in a country that ...	25	140	5240.0
4	Where can I change my clothes at the airport?	16	128	12020.0
5	Boss is asking for passport, but it has a stam...	24	121	54982.0
6	How to avoid toddlers on a long-distance plane...	19	120	24955.0
7	Job interview in London requires me to wire mo...	23	116	14827.0
8	What to do without underwear on a 4 day trip?	13	110	11713.0
9	OK, we are all adults here, so what is a bidet...	28	109	52265.0

Teraz zajmiemy się ramkę dannych, która jest stworzona za pomocą funkcji z pakiety *pandas*. Najpierw stworzymy pomocniczą ramkę dannych CmtTotScr:

In [18]:

```
# za pomocą fillna(-1) zapewniamy sobie aggregacje według wartości NA
CmtTotScr = Comments.loc[:, ["PostId", "UserId", "Score"]]\
            .fillna(-1)\
            .groupby(["PostId", "UserId"])\
            .agg(np.sum)\
            .reset_index()

CmtTotScr.columns = ["PostId", "UserId", "CommentsTotalScore"] # Zmieniamy nazwy kolumn
```

Teraz przejdziemy do wynikowej ramki danych:

In [16]:

```
# tworzymy ramkę newPosts z potrzebnymi kolumnami z Posts
newPosts = Posts.loc[:, ["Title", "CommentCount", "ViewCount", "Id", "PostTypeId"]]

# łączymy CmtTotScr i newPosts
df_pd_2 = CmtTotScr.merge(newPosts, left_on = "PostId", right_on = "Id")
# wybieramy tylko te wiersze, w których wartość PostTypeId jest równa się 1
df_pd_2 = df_pd_2.loc[df_pd_2.PostTypeId == 1, :]
# usuwamy niepotrzebne kolumny, sortujemy malejąco odnośnie kolumny CommentsTotalScore,
# zmieniamy kolejność kolumn, ustawiamy kolejność wierszy od 0 i wypisujemy pierwsze 10 wierszy
df_pd_2 = df_pd_2.drop(df_pd_2.columns[[0, 1, 6, 7]], axis = 1).sort_values("CommentsTotalScore", ascending = False)\
                                    .iloc[:, [1, 2, 0, 3]]\
                                    .reset_index(drop = True)\
                                    .head(10)

df_pd_2
```

Out[16]:

	Title	CommentCount	CommentsTotalScore	ViewCount
0	Boss is asking for passport, but it has a stam...	24	207	54982.0
1	Why don't airlines have backup planes just in ...	26	172	14516.0
2	OK we're all adults here, so really, how on ea...	27	155	73808.0
3	How to cross a road by foot in a country that ...	25	140	5240.0
4	Where can I change my clothes at the airport?	16	128	12020.0
5	Boss is asking for passport, but it has a stam...	24	121	54982.0
6	How to avoid toddlers on a long-distance plane...	19	120	24955.0
7	Job interview in London requires me to wire mo...	23	116	14827.0
8	What to do without underwear on a 4 day trip?	13	110	11713.0
9	OK, we are all adults here, so what is a bidet...	28	109	52265.0

Sprawdzimy czy te dwie ramki danych są równoważne za pomocą funkcji *equals()*:

In [17]:

```
df_sql_2.equals(df_pd_2)
```

Out[17]:

```
True
```

Otrzymujemy wartość True, więc ramki danych są identyczne.

Rozdział IV

Podsumowanie

Rozwiązaliśmy dwa zapytania w języku SQL za pomocą fynkji bazowych Python praz funkcji z pakietu **pandas**. Otrzymaliśmy we wszystkich dwóch sprawdzaniach równaważności funkcji wartość True, czyli ramki danych różnią się tylko sposobem implementacji.

Moim zdaniem, rozwiązania za pomocą **pandas.read_sql_query** są czytelniejsze dla użytkowników. Ale z drugiej strony, funkcje z pakietu **pandas** zapewniają nam szybsze rozwiązania zapytań (nie mierzyliśmy czas wykonania funkcji, ale w internecie jest dużo informacji na ten temat, na przykład https://www.reddit.com/r/SQL/comments/hemr21/is_pandas_or_sql_faster/).

Porównując rozwiązania w funkcjach **pandas** i rozwiązania w funkcjach pakietów **data.table** oraz **dplyr** języka R, moim zdaniem implementacja zapytań w **pandas** jest bardzo podobna do **dplyr**, szybkość wykonywania zapytań (na podstawie informacji z internetu) w **pandas** jest prawie taka sama jak i w **data.table**.

Zauważyłam, że pakiet **pandas** cieszy się popularnością według użytkowników w porównaniu do pakietów **data.table** oraz **dplyr**.

In [ ]: