PADR 2019/2020

Praca domowa nr 2 (max. = 25 p.)

Maksymalna ocena: 25 p. (7 zadań po max. 3.5p. oraz 0.5p. za ogólną postać raportu)

Termin oddania pracy: 04.12.2019 r., godz. 06:00

Do przesłania na adres `A.Geras@mini.pw.edu.pl` ze swojego konta pocztowego `*@*pw.edu.pl`:

- `Nick_Nazwisko_Imie_NrAlbumu_pd2.Rmd` (rozwiązanie zadań)
- `Nick_Nazwisko_Imie_NrAlbumu_pd2.html` (skompilowana wersja powyższego).

Temat wiadomości: `[PADR-1920] Praca domowa nr 2`.

1 Zbiory danych

Będziemy pracować na uproszczonym zrzucie zanonimizowanych danych z serwisu https://bicycles.stackexchange.com/ (na marginesie: pełen zbiór danych dostępny jest pod adresem https://archive.org/details/stackexchange), który składa się z następujących ramek danych:

- Badges.csv
- Comments.csv
- PostLinks.csv
- Posts.csv
- Tags.csv
- Users.csv
- Votes.csv

Przykładowe wywołanie — ładowanie zbioru `Tags`:

```
options(stringsAsFactors=FALSE)
# ww. pliki pobralismy do katalogu travel_stackexchange_com/
Tags <- read.csv("travel_stackexchange_com/Tags.csv.gz")
head(Tags)
```

Przed przystąpieniem do rozwiązywania zadań zapoznaj się z ww. serwisem oraz znaczeniem poszczególnych kolumn we wspomnianych ramkach danych, zob. http://www.gagolewski.com/resources/data/travel_stackexchange_com/readme.txt.

2 Informacje ogólne

Rozwiąż poniższe zadania przy użyciu wywołań funkcji bazowych oraz tych, które udostępniają pakiety `dplyr` oraz `data.table` – nauczysz się ich samodzielnie; ich dokumentację znajdziesz łatwo w internecie. Każdemu z 7 poleceń SQL powinny odpowiadać cztery równoważne sposoby ich implementacji w R, kolejno:

1. `sqldf::sqldf()`;
2. tylko funkcje bazowe;
3. `dplyr`;
4. `data.table`.

Upewnij się, że zwracane wyniki są ze sobą tożsame (ewentualnie z dokładnością do permutacji wierszy wynikowych ramek danych, zob. np. funkcję `dplyr::all_equal`). W każdym przypadku należy podać słowną (opisową) interpretację każdego zapytania.

Ponadto w każdym przypadku porównaj czasy wykonania napisanych przez Ciebie wyrażeń przy użyciu jednego wywołania `microbenchmark::microbenchmark()`, np.:

```
microbenchmark::microbenchmark(
   sqldf=rozwiazanie_sqldf,
   base=rozwiazanie_bazowe,
   dplyr=rozwiazanie_dplyr,
   data.table=rozwiazanie_datatable
)
```

Wszystkie rozwiązania umieść w jednym (estetycznie sformatowanym) raporcie knitr/Markdown. Za bogate komentarze do kodu, dyskusję i ewentualne rozwiązania alternatywne można otrzymać max. 0.5 p.

3 Zadania do rozwiązania

```
--- Zadanie 1
SELECT PostId, COUNT(*) AS UpVotes
FROM Votes
WHERE VoteTypeId=2
GROUP BY PostId
```

```
--- Zadanie 2
SELECT Title, Score, ViewCount, FavoriteCount
FROM Posts
WHERE PostTypeId=1 AND FavoriteCount >= 25 AND ViewCount>=10000
```

```
--- Zadanie 3
SELECT Tags.TagName, Tags.Count, Posts.OwnerUserId,
        Users.Location, Users.DisplayName
FROM Tags
JOIN Posts ON Posts.Id=Tags.WikiPostId
JOIN Users ON Users.AccountId=Posts.OwnerUserId
WHERE OwnerUserId != -1
ORDER BY Count DESC LIMIT 10
```

```
--- Zadanie 4
SELECT Posts.Title, RelatedTab.NumLinks FROM
    (SELECT RelatedPostId AS PostId, COUNT(*) AS NumLinks
     FROM PostLinks GROUP BY RelatedPostId) AS RelatedTab
JOIN Posts ON RelatedTab.PostId=Posts.Id
WHERE Posts.PostTypeId=1
ORDER BY NumLinks DESC LIMIT 10
```

```
--- Zadanie 5
SELECT UpVotesTab.*, Posts.Title FROM
(
    SELECT PostId, COUNT(*) AS UpVotes FROM Votes WHERE VoteTypeId=2 GROUP BY PostId
) AS UpVotesTab
JOIN Posts ON UpVotesTab.PostId=Posts.Id
WHERE Posts.PostTypeId=1
ORDER BY UpVotesTab.UpVotes DESC LIMIT 10
```

```
--- Zadanie 6
SELECT UpVotesTab.PostId, UpVotesTab.UpVotes, IFNULL(DownVotesTab.DownVotes, 0) AS DownVotes
    FROM
```

```
        (
            SELECT PostId, COUNT(*) AS UpVotes FROM Votes
                WHERE VoteTypeId=2 GROUP BY PostId
        ) AS UpVotesTab
    LEFT JOIN
        (
            SELECT PostId, COUNT(*) AS DownVotes FROM Votes
                WHERE VoteTypeId=3 GROUP BY PostId
        ) AS DownVotesTab
    ON UpVotesTab.PostId=DownVotesTab.PostId
```

```
--- Zadanie 7
SELECT PostId, UpVotes-DownVotes AS Votes FROM (
    SELECT UpVotesTab.PostId, UpVotesTab.UpVotes, IFNULL(DownVotesTab.DownVotes, 0) AS DownVotes
        FROM
            (
                SELECT PostId, COUNT(*) AS UpVotes FROM Votes
                    WHERE VoteTypeId=2 GROUP BY PostId
            ) AS UpVotesTab
        LEFT JOIN
            (
                SELECT PostId, COUNT(*) AS DownVotes
                    FROM Votes WHERE VoteTypeId=3 GROUP BY PostId
            ) AS DownVotesTab
        ON UpVotesTab.PostId=DownVotesTab.PostId

    UNION

    SELECT DownVotesTab.PostId, IFNULL(UpVotesTab.UpVotes, 0) AS UpVotes, DownVotesTab.DownVotes
        FROM
            (
                SELECT PostId, COUNT(*) AS DownVotes FROM Votes
                    WHERE VoteTypeId=3 GROUP BY PostId
            ) AS DownVotesTab
        LEFT JOIN
            (
                SELECT PostId, COUNT(*) AS UpVotes FROM Votes
                    WHERE VoteTypeId=2 GROUP BY PostId
            ) AS UpVotesTab
        ON DownVotesTab.PostId=UpVotesTab.PostId
)
```