PDU 2022/2023

Praca domowa (sprawdzian nr 3) (max. = 40 p.)

Maksymalna ocena: 40 p.

Prace domowe należy przesłać za pośrednictwem platformy `Moodle` – **jeden plik .R** o nazwie `Imie_Nazwisko_nrAlbumu_PD3.R`.

Plik powinien zawierać rozwiązanie zadań zgodne z załączonym szablonem. Uwaga: nazwy plików nie powinny zawierać polskich liter diakrytyzowanych (przekształć $ą \to a$ itd.).

1 Zbiory danych

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

- Posts.csv.gz
- Users.csv.gz
- Comments.csv.gz

Przed przystąpieniem do rozwiązywania zadań zapoznaj się z ww. serwisem oraz znaczeniem poszczególnych kolumn we wspomnianych ramkach danych, zob. https://ia600107.us.archive.org/27/items/stackexchange/readme.txt

Przykładowe wywołanie — ładowanie zbioru `Tags`:

```
# ww. pliki pobralismy do katalogu travel_stackexchange_com/
Posts <- read.csv("travel_stackexchange_com/Posts.csv.gz")
head(Posts)
```

Uwaga: Nazwy ramek danych po wczytaniu zbiorów powinny wyglądać następująco: `Comments`, `Posts` oraz `Users`.

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 **5** poleceń SQL powinny odpowiadać cztery równoważne sposoby ich implementacji w R, kolejno:

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

Rozwiązanie każdego zadania powinno być zaimplementowane jako funkcje: `sql_i()`, `base_i()`, `dplyr_i()`, `table_i()`, o nazwach i parametrach określonych w szablonie.

W przypadku każdego zadania:

1. Upewnij się, że zwracane wyniki są ze sobą tożsame (ewentualnie z dokładnością do permutacji wierszy wynikowych ramek danych, zob. np. funkcję `dplyr::all_equal` lub `compare::compare`).

2. Kod rozwiązań opatrz komentarzami oraz podaj słowną interpretację (tzn. intuicyjne – „dla laika'' – tłumaczenie) każdego zapytania.
3. W każdym przypadku porównaj czasy wykonania napisanych przez Ciebie wyrażeń przy użyciu jednego wywołania `microbenchmark::microbenchmark()`, np.:

```
microbenchmark::microbenchmark(
   sqldf = sql_i(...),
   base = base_i(...),
   dplyr = dplyr_i(...),
   data.table = table_i(...)
```

UWAGA

Wysyłając rozwiązanie upewnij się, że plik jest zgodny z szablonem rozwiązania, tzn. nazwy funkcji i ich parametrów nie zostały zmienione oraz wskazane fragmenty kodu zostały zakomentowane.

3 Zadania do rozwiązania

```
--- 1)
SELECT Location, SUM(UpVotes) as TotalUpVotes
FROM Users
WHERE Location != ''
GROUP BY Location
ORDER BY TotalUpVotes DESC
LIMIT 10
```

```
--- 2)
SELECT STRFTIME('%Y', CreationDate) AS Year, STRFTIME('%m', CreationDate) AS Month,
       COUNT(*) AS PostsNumber, MAX(Score) AS MaxScore
FROM Posts
WHERE PostTypeId IN (1, 2)
GROUP BY Year, Month
HAVING PostsNumber > 1000
```

```
--- 3)
SELECT Id, DisplayName, TotalViews
FROM (
        SELECT OwnerUserId, SUM(ViewCount) as TotalViews
        FROM Posts
        WHERE PostTypeId = 1
        GROUP BY OwnerUserId
     ) AS Questions
JOIN Users
ON Users.Id = Questions.OwnerUserId
ORDER BY TotalViews DESC
LIMIT 10
```

```sql
--- 4)
SELECT DisplayName, QuestionsNumber, AnswersNumber, Location, Reputation, UpVotes, DownVotes
FROM (
        SELECT *
        FROM (
                SELECT COUNT(*) as AnswersNumber, OwnerUserId
                FROM Posts
                WHERE PostTypeId = 2
                GROUP BY OwnerUserId
             ) AS Answers
        JOIN
             (
                SELECT COUNT(*) as QuestionsNumber, OwnerUserId
                FROM Posts
                WHERE PostTypeId = 1
                GROUP BY OwnerUserId
              ) AS Questions
        ON Answers.OwnerUserId = Questions.OwnerUserId
        WHERE AnswersNumber > QuestionsNumber
        ORDER BY AnswersNumber DESC
        LIMIT 5
    ) AS PostsCounts
JOIN Users
ON PostsCounts.OwnerUserId = Users.Id
```

```sql
--- 5)
SELECT Title, CommentCount, ViewCount, CommentsTotalScore, DisplayName, Reputation, Location
FROM (
        SELECT Posts.OwnerUserId, Posts.Title, Posts.CommentCount, Posts.ViewCount,
               CmtTotScr.CommentsTotalScore
        FROM (
                SELECT PostId, SUM(Score) AS CommentsTotalScore
                FROM Comments
                GROUP BY PostId
             ) AS CmtTotScr
        JOIN Posts ON Posts.Id = CmtTotScr.PostId
        WHERE Posts.PostTypeId=1
    ) AS PostsBestComments
JOIN Users ON PostsBestComments.OwnerUserId = Users.Id
ORDER BY CommentsTotalScore DESC
LIMIT 10
```