PDRPy 2023/2024

Praca domowa nr 1 (max. = 40 p.)

Maksymalna ocena: 40 p.

Do przesłania przy użyciu platformy `LeON` następujące pliki:

- `Nazwisko_Imie_NrAlbumu_pd1.R` - rozwiązania zadań (zgdone z załączonym szabolonem);
- `Nazwisko_Imie_NrAlbumu_pd1.Rmd` - raport w `Markdown/knitr`;
- `Nazwisko_Imie_NrAlbumu_pd1.html` - skompilowana wersja powyższego.
  - Wynikowy raport (`.Rmd`, `.html`) powinien zawierać sprawdzenie równoważności wyników uzyskanych przy pomocy każdej metody, pomiar i ocenę czasu wykonania oraz interpretację zapytań (zob. Informacje Ogólne).
  - Raport powinien także zawierać krótki wstęp (1-2 zdania), w tym wczytanie danych, oraz podsumowanie.

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
- PostLinks.csv.gz

Przed przystąpieniem do rozwiązywania zadań zapoznaj się z omawianym serwisem i strukturą zbiorów danych (np. jakie informacje reprezentują poszczególne kolumny) https://archive.org/27/items/stackexchange/readme.txt.

Przykładowe wywołanie – ładowanie zbioru `Posts`:

```
# ww. pliki znajdują się w katalogu travel_stackexchange_com
Posts <- read.csv("travel_stackexchange_com/Posts.csv.gz")
head(Posts)
```

Uwaga:

1. **Nazwy ramek danych po wczytaniu zbiorów powinny wyglądać następująco:** `Posts`, `Users`, `Comments`, `PostLinks`.
2. **W przypadku, gdy tworzą Państwo ramki danych o typie `data.table` powinny mieć one nazwy:** `PostsDT`, `UsersDT`, `CommentsDT`, `PostLinksDT`.

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 łatwo odnajdziesz w internecie. Każdemu z **5** poleceń SQL powinny odpowiadać cztery równoważne sposoby ich implementacji w R, kolejno:

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

W raporcie:

1. Koniecznie upewnij się, że zwracane wyniki są ze sobą tożsame (z dokładnością do permutacji wierszy wynikowych ramek danych).
2. W w każdym przypadku porównaj czasy wykonania napisanych przez Ciebie wyrażeń przy użyciu jednego wywołania `microbenchmark::microbenchmark()`, np.:

```
microbenchmark::microbenchmark(
   sqldf = ...,
   base = ...,
   dplyr = ...,
   data.table = ...
)
```

3. Ponadto w każdym przypadku należy podać słowną („dla laika”) interpretację każdego zapytania.

Łączna ocena każdego z 5. zadań to 7 pkt (za poszczególne komponenty umieszczone w pliku `.R` oraz raporcie: rozwiązanie i sprawdzenie równoważności wyników, pomiar i ocena czasu wykonania, opis słowny zapytań) oraz 5 pkt za ogólną postać raportu (np. komentarze, wstęp, podsumowanie).

3 Zadania do rozwiązania

```
--- 1)
SELECT Location, COUNT(*) AS Count
FROM (
    SELECT Posts.OwnerUserId, Users.Id, Users.Location
    FROM Users
    JOIN Posts ON Users.Id = Posts.OwnerUserId
)
WHERE Location NOT IN ('')
GROUP BY Location
ORDER BY Count DESC
LIMIT 10
```

```
--- 2)
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
```

```sql
--- 3)
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
```

```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
    Users.AccountId,
    Users.DisplayName,
    Users.Location,
    AVG(PostAuth.AnswersCount) as AverageAnswersCount
FROM
(
    SELECT
        AnsCount.AnswersCount,
        Posts.Id,
        Posts.OwnerUserId
    FROM (
            SELECT Posts.ParentId, COUNT(*) AS AnswersCount
            FROM Posts
            WHERE Posts.PostTypeId = 2
            GROUP BY Posts.ParentId
          ) AS AnsCount
    JOIN Posts ON Posts.Id = AnsCount.ParentId
) AS PostAuth
JOIN Users ON Users.AccountId=PostAuth.OwnerUserId
GROUP BY OwnerUserId
ORDER BY AverageAnswersCount DESC
LIMIT 10
```