DPRPy 2022/2023

Homework assignment no. 1 (max. = 40 p.)

Maximum grade: 40 p.

Deadline: **28.11.2021, 23:59** (28 days = 4 weeks).

Homework should be sent via the `Moodle` platform as follows. You should send **exactly 3 files**:

1. `Last-name_First-name_assignment_1.R` - an R script containing solutions to tasks (prepared according to the attached template);
2. `Last-name_First-name_assignment_1.Rmd` a report prepared with `Markdown` / `knitr` containing :

```
source('Last-name_First-name_assignment_1.R')
```

- attachment of packages,
- reading the data,
- results of comparing the equivalence of solutions for each task,
- measurements of execution times,
- queries interpretation.

3. `Last-name_First-name_assignment_1.html` - compiled to HTML version of the above.

1 Data description

We are working on a simplified dump of anonymised data from the website https://travel.stackexchange.com/ (by the way: full data set is available at https://archive.org/details/stackexchange), which consists of the following data frames:

- Badges.csv.gz
- Comments.csv.gz
- Posts.csv.gz
- Users.csv.gz
- Votes.csv.gz

Before starting to solve the problems familiarize yourself with the said service and data sets structure (e.g. what information individual columns represent), see https://archive.org/27/items/stackexchange/readme.txt.

Example: loading the set `Posts`:

```
options(stringsAsFactors=FALSE)
# if files are saved at "travel_stackexchange_com/" directory
Posts <- read.csv("travel_stackexchange/Posts.csv.gz")
head(Posts)
```

2 Tasks description

Solve the following tasks using base functions calls and those provided by the `dplyr` and`data.table` packages - you will learn them on your own; their documentation (and tutorials) is easy to find online. Each of the **5 SQL queries** should have four implementations in `R`:

1. `sqldf::sqldf()` – reference solution;

2. only base functions (1.5 p.);
3. `dplyr` (1.5 p.);
4. `data.table` (1.5 p.).

Make sure that the obtained results are equivalent (possibly with row permutation accuracy; up to 1 p. for each task). You can propose a function that implements relevant tests (e.g. based on `compare::compare()` or `dplyr::all_equal()`) - the results of such comparisons should be included in the final report. In addition, compare the execution times written by you in each case using one call to `microbenchmark :: microbenchmark ()` (1 p.), e.g .:

```
microbenchmark::microbenchmark(
   sqldf=sqldf_solution,
   base=base_functions_solution,
   dplyr=dplyr_solutions,
   data.table=datatable_solution
)
```

In addition, in each case, it is necessary to provide "intuitive" interpretation of each query (0.5 p.).

Be sure to format `knitr` / `Markdown` report nicely. For rich code comments, discussion and possible alternative solutions you can obtained max. 5 p.

The solutons code **should not** be included in the report.

3 SQL queries

```
--- 1)
SELECT STRFTIME('%Y', CreationDate) AS Year, COUNT(*) AS TotalNumber
FROM Posts
GROUP BY Year
```

```
--- 2)
SELECT Id, DisplayName, SUM(ViewCount) AS TotalViews
FROM Users
JOIN (
        SELECT OwnerUserId, ViewCount FROM Posts WHERE PostTypeId = 1
     ) AS Questions
ON Users.Id = Questions.OwnerUserId
GROUP BY Id
ORDER BY TotalViews DESC
LIMIT 10
```

```
--- 3)
ELECT Year, Name, MAX((Count * 1.0) /  CountTotal) AS MaxPercentage
FROM (
        SELECT BadgesNames.Year, BadgesNames.Name, BadgesNames.Count, BadgesYearly.CountTotal
        FROM (
                SELECT Name, COUNT(*) AS Count, STRFTIME('%Y', Badges.Date) AS Year
                FROM Badges
                GROUP BY Name, Year
             ) AS BadgesNames
        JOIN (
                SELECT COUNT(*) AS CountTotal, STRFTIME('%Y', Badges.Date) AS Year
                FROM Badges
                GROUP BY YEAR
             ) AS BadgesYearly
```

```
        ON BadgesNames.Year = BadgesYearly.Year
)
GROUP BY Year
```

```
--- 4)
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
```

```
--- 5)
SELECT Posts.Title, STRFTIME('%Y-%m-%d', Posts.CreationDate) AS Date, VotesByAge.*
FROM Posts
JOIN (
        SELECT PostId,
               MAX(CASE WHEN VoteDate = 'before' THEN Total ELSE 0 END) BeforeCOVIDVotes,
               MAX(CASE WHEN VoteDate = 'during' THEN Total ELSE 0 END) DuringCOVIDVotes,
               MAX(CASE WHEN VoteDate = 'after' THEN Total ELSE 0 END) AfterCOVIDVotes,
               SUM(Total) AS Votes
        FROM (
                SELECT PostId,
                CASE STRFTIME('%Y', CreationDate)
                    WHEN '2022' THEN 'after'
                    WHEN '2021' THEN 'during'
                    WHEN '2020' THEN 'during'
                    WHEN '2019' THEN 'during'
                    ELSE 'before'
                END VoteDate, COUNT(*) AS Total
                FROM Votes
                WHERE VoteTypeId IN (3, 4, 12)
                GROUP BY PostId, VoteDate
             ) AS VotesDates
        GROUP BY VotesDates.PostId
    ) AS VotesByAge ON Posts.Id = VotesByAge.PostId
WHERE Title NOT IN ('') AND DuringCOVIDVotes > 0
ORDER BY DuringCOVIDVotes DESC, Votes DESC
LIMIT 20
```