DPRPy 2023/2024

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

Maximum grade: 40 p.

Deadline: **23.11.2023, 11:59** (28 days = 4 weeks).

Homework should be sent via the `LeOn` 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` **or `.ipynb` file prepared with `Jupyter Notebook`** 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 (in `Jupyter Notebook` use `Download` option).

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:

- Posts.csv.gz
- Users.csv.gz
- Comments.csv.gz
- PostLinks.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) # needed only for R < 4.0
# 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 (you can ignore row permutation; up to 1 p. for each task). You can propose a function that implements relevant comparisons or use implementation available in `R` (e.g. `compare::compare()` or `dplyr::all_equal()`). The results of such comparisons should be included in the final report. In addition, compare the execution times of your solutions 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 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
    Questions.Id,
    Questions.Title,
    BestAnswers.MaxScore,
    Posts.Score AS AcceptedScore,
    BestAnswers.MaxScore-Posts.Score AS Difference
FROM (
        SELECT Id, ParentId, MAX(Score) AS MaxScore
        FROM Posts
        WHERE PostTypeId==2
        GROUP BY ParentId
     ) AS BestAnswers
JOIN (
        SELECT * FROM Posts
        WHERE PostTypeId==1
     ) AS Questions
ON Questions.Id=BestAnswers.ParentId
JOIN Posts ON Questions.AcceptedAnswerId=Posts.Id
WHERE Difference>50
ORDER BY Difference DESC
```