DPRPy 2021/2022

Homework assigment no. 3 (max. = 25 p.)

Maximum grade: 25 p.

Deadline: 25.01.2022, 23.59

Homework should be sent via the `Moodle` platform - **one archive `.zip`**[1] named

`Last-name_First-name_assgment_3.zip`

(one directory inside: `Last-name_First-name_assgment_3`), in which the following files will be placed:

- `Last-name_First-name_assgment_3.ipynb` (report prepared with `Jupyter` / `Markdown` containing task solutions, comments, etc.),
- `Last-name_First-name_assgment_3.html` (HTML version of the above - see Download options in Jupyter notebooks).

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
- PostLinks.csv.gz
- Posts.csv.gz
- Tags.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 `Tags`:

```
import pandas as pd
import numpy as np

Tags = pd.read_csv("travel_stackexchange_com/Tags.csv.gz",
                   compression = "gzip")
```

[1]So not: .rar, .7z etc.

2 Tasks description

Solve the following tasks using `pandas` methods and functions. Each of the **3 SQL queries** should have two implementations in `Python`:

1. `pandas.read_sql_query("""zapytanie SQL""")` - reference solution;
2. calling methods and functions from `pandas` package (3 p.).

Make sure that the obtained results are equivalent (possibly with an accuracy of the row permutation of the result data frames), e.g., see the `.equals()` method from the `pandas` package. The results of such comparision should be included in the final report (1.5 p. for each task).

Put all solutions in one (nicely formatted) Jupyter notebook (use `Markdown` option) report. For rich code comments, discussion and possible alternative solutions you can obtained max. 2.5 p.

2.1 Data Base

You can work with the database in the following way:

```
import os, os.path
import sqlite3
import tmpfile

# path to database file
baza = os.path.join(tempfile.mkdtemp(), 'example.db')
if os.path.isfile(baza): # if this file already exists...
    os.remove(baza)      # ...we will remove it

conn = sqlite3.connect(baza)      # create the connection

Badges.to_sql("Badges", conn)     # import the data frame into the database
Comments.to_sql("Comments", conn)
PostLinks.to_sql("PostLinks", conn)
Posts.to_sql("Posts", conn)
Tags.to_sql("Tags", conn)
Users.to_sql("Users", conn)
Votes.to_sql("Votes", conn)

#
pd.read_sql_query("""
                  SQL query
                  """, conn)
# ...
# tasks solution
# after finishing work, we close the connection
#
conn.close()
```

3 SQL queries

```
--- 1)
SELECT
```

```
    Name,
    COUNT(*) AS Number,
    MIN(Class) AS BestClass
FROM Badges
GROUP BY Name
ORDER BY Number DESC
LIMIT 10
```

```
--- 2)
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
```

```
--- 3)
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
```

```sql
--- 4)
SELECT
    Posts.Title,
    UpVotesPerYear.Year,
    MAX(UpVotesPerYear.Count) AS Count
FROM (
        SELECT
            PostId,
            COUNT(*) AS Count,
            STRFTIME('%Y', Votes.CreationDate) AS Year
        FROM Votes
        WHERE VoteTypeId=2
        GROUP BY PostId, Year
    ) AS UpVotesPerYear
JOIN Posts ON Posts.Id=UpVotesPerYear.PostId
WHERE Posts.PostTypeId=1
GROUP BY Year
ORDER BY Year ASC
```

```sql
--- 5)
SELECT
    Posts.Title,
    VotesByAge2.OldVotes
FROM Posts
JOIN (
    SELECT
        PostId,
        MAX(CASE WHEN VoteDate = 'new' THEN Total ELSE 0 END) NewVotes,
        MAX(CASE WHEN VoteDate = 'old' THEN Total ELSE 0 END) OldVotes,
        SUM(Total) AS Votes
    FROM (
        SELECT
            PostId,
            CASE STRFTIME('%Y', CreationDate)
                WHEN '2021' THEN 'new'
                WHEN '2020' THEN 'new'
                ELSE 'old'
                END VoteDate,
            COUNT(*) AS Total
        FROM Votes
        WHERE VoteTypeId IN (1, 2, 5)
        GROUP BY PostId, VoteDate
    ) AS VotesByAge
    GROUP BY VotesByAge.PostId
    HAVING NewVotes=0
) AS VotesByAge2 ON VotesByAge2.PostId=Posts.ID
WHERE Posts.PostTypeId=1
ORDER BY VotesByAge2.OldVotes DESC
LIMIT 10
```