DPRPy 2022/2023

Homework assignment no. 2 (max. = 25 p.)

Maximum grade: 25 p.

Deadline: **19.12.2022, 23:59** (14 days = 2 weeks).

Homework should be sent via the `Moodle` platform as follows. You should send **exactly 2 files**:

1. `Last-name_First-name_assignment_1.py` - an Python script containing solutions to tasks (prepared according to the attached template);
2. `Last-name_First-name_assignment_1.ipynb` a `Jupyter` notebook containing :

```
import numpy as np
import pandas as pd
from Last-name_First-name_assignment_1 import *
```

- reading the data,
- results of comparing the equivalence of solutions for each task.

1 Data description

Note: Use the data (i.e. csv files) from Homework Assignment no. 1

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`:

```
import pandas as pd
import numpy as np

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

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).

Remember to format your Jupyter notebook (use `Markdown` option) nicely, i.e., use sections / subsections in order to highlight each task, include title and short summary (one two sentences). This will be worth 2.5 p.

2.1 Data Base

You can work with the database in the following way:

```
import os, os.path
import sqlite3
import tempfile

# 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 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
```

```sql
--- 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
```