DPRPy 2023/2024

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

Maximum grade: 25 p.

Deadline: **18.12.2023, 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_2.py` - an Python script containing solutions to tasks (prepared according to the attached template);
2. `Last-name_First-name_assignment_2.ipynb` a `Jupyter` notebook containing :

```
import numpy as np
import pandas as pd
from Last-name_First-name_assignment_2 import *
```

- reading the data,
- creation of the database (see exemplary code below),
- 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:

- 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`:

```
import pandas as pd
import numpy as np

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

2 Tasks description

Solve the following tasks using `pandas` methods and functions. Each of the 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 comparison 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 Database

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

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

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

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
```

```
--- 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
```