PDRPy 2023/2024

Praca domowa nr 2 (max. = 25 p.)

Maksymalna ocena: 25 p.

Do przesłania przy użyciu platformy `LeON` następujące pliki:

- `Nazwisko_Imie_PD2.py` - rozwiązania zadań według szablonu;
- `Nazwisko_Imie_PD2.ipynb` - raport w `Jupyter Notebook` gdzie znajduje się wczytanie danych oraz porównanie wyników.

  Koniecznie pamiętaj o zaimportowaniu rozwiązań z pliku .py, tzn.:

  ```
  from Nazwisko_Imie_PD2 import *
  ```

1 Zbiory danych

Będziemy pracować na uproszczonym zrzucie zanonimizowanych danych z serwisu https://travel.stackexchange.com/ (na marginesie: pełen zbiór danych dostępny jest pod adresem https://archive.org/details/stackexchange), który składa się z następujących ramek danych:

- Posts.csv.gz
- Users.csv.gz
- Comments.csv.gz
- PostLinks.csv.gz

Przed przystąpieniem do rozwiązywania zadań zapoznaj się z omawianym serwisem i strukturą zbiorów danych (np. jakie informacje reprezentują poszczególne kolumny) https://archive.org/27/items/stackexchange/readme.txt.

Przykładowe wywołanie – ładowanie zbioru `Posts`:

```
import pandas as pd
import numpy as np

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

Każdą z ramek danych należy wyeksportować do bazy danych SQLite przy użyciu wywołania metody `to_sql()` w klasie `pandas.DataFrame`. Dokładniej, pracę z bazą danych możemy przeprowadzić w następujący sposób.

```
import os, os.path
import sqlite3

# sciezka dostępu do bazy danych:
baza = os.path.join('przyklad.db')
if os.path.isfile(baza): # jesli baza już istneje...
    os.remove(baza)      # ...usuniemy ja
                         # zeby miec zaczac z ,,czystą kartą''
```

```
conn = sqlite3.connect(baza)      # połączenie do bazy danych

Badges.to_sql("Badges", conn)     # importujemy ramkę danych do bazy danych
# ....

#
pd.read_sql_query("""
                  Zaptanie SQL
                  """, conn)
# ...
# rozwiązania zadan
# po skończonej pracy zamykamy połączenie
#
conn.close()
```

W szczególności należy zagwarantować, że w każdym przypadku wynik jest klasy `DataFrame` a nie `Series`.

Uwaga:

Nazwy ramek danych po wczytaniu zbiorów powinny wyglądać następująco: `Comments`, `Posts`, `Users`, `PostLinks`.

2 Informacje ogólne

Rozwiąż poniższe zadania przy użyciu wywołań funkcji i metod z pakietu `pandas`. Każdemu z **5** poleceń SQL powinny odpowiadać dwa równoważne sposoby ich implementacji, kolejno:

1. wywołanie `pandas.read_sql_query("""zapytanie SQL""")`;
2. wywołanie ciągu „zwykłych” metod i funkcji z pakietu `pandas`.

Upewnij się, że zwracane wyniki są ze sobą tożsame (ewentualnie z dokładnością do permutacji wierszy wynikowych ramek danych), por. np. metodę `.equals()` z pakietu `pandas`.

Wszystkie rozwiązania umieść w jednym (estetycznie sformatowanym) raporcie `Jupyter` (za który – tj. za komentarze do kodu, dyskusję, format można uzyskać max. 5 p.).

3 Zadania do rozwiązania

```
--- 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
```

```sql
--- 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
    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
```