# PDU 2023/2024

Praca domowa nr 3 (max. = 15 p.)

Prace domowe należy przesłać za pośrednictwem platformy `LeON` przygotowany zgodnie z szablonem[1] plik z rozwiązanianimi.

## 1 Zbiory danych

Ponownie będziemy pracować na uproszczonym zrzucie zanonimizowanych danych z serwisu https://travel.stackexchange.com/, który składa się z następujących ramek danych:

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

**Uwaga: wykorzystujemy ramki danych z pracy domowej nr 1.**

Przed przystąpieniem do rozwiązywania zadań przypomnij sobie znaczenie poszczególnych kolumn we wspomnianych ramkach danych, zob. https://ia600107.us.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 pzeprowadzić w następujący sposób.

```
import os, os.path
import sqlite3

baza = 'przyklad.db'              # sciezka dostępu do bazy danych:

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

Comments.to_sql("Comments", conn) # importujemy ramkę danych do bazy danych
Posts.to_sql("Posts", conn)
Users.to_sql("Users", conn)
# etc.
#
```

[1] Szablony dostępne są na `LeON`.

```
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:** `Badges`, `Comments`, `Tags`, `Posts`, `Users`, `Votes`, `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 i kolumn wynikowych ramek danych), por. np. metodę `.equals()` z pakietu `pandas`.

## 3 Zadania do rozwiązania

```
--- 1)
SELECT STRFTIME('%Y', CreationDate) AS Year,
       STRFTIME('%m', CreationDate) AS Month,
       COUNT(*) AS TotalAccountsCount,
       AVG(Reputation) AS AverageReputation
FROM Users
GROUP BY Year, Month
```

```
--- 2)
SELECT Users.DisplayName, Users.Location, Users.Reputation,
       STRFTIME('%Y-%m-%d', Users.CreationDate) AS CreationDate,
       Answers.TotalCommentCount
FROM (
        SELECT OwnerUserId, SUM(CommentCount) AS TotalCommentCount
        FROM Posts
        WHERE PostTypeId == 2 AND OwnerUserId != ''
        GROUP BY OwnerUserId
     ) AS Answers
JOIN Users ON Users.Id == Answers.OwnerUserId
ORDER BY TotalCommentCount DESC
LIMIT 10
```

```sql
--- 3)
SELECT Spam.PostId, UsersPosts.PostTypeId, UsersPosts.Score,
       UsersPosts.OwnerUserId, UsersPosts.DisplayName,
       UsersPosts.Reputation
FROM (
        SELECT PostId
        FROM Votes
        WHERE VoteTypeId == 12
     ) AS Spam
JOIN (
        SELECT Posts.Id, Posts.OwnerUserId, Users.DisplayName,
               Users.Reputation, Posts.PostTypeId, Posts.Score
        FROM Posts JOIN Users
        ON Posts.OwnerUserId = Users.Id
     ) AS UsersPosts
ON Spam.PostId = UsersPosts.Id
```

```sql
--- 4)
SELECT Users.Id, Users.DisplayName, Users.UpVotes, Users.DownVotes, Users.Reputation,
       COUNT(*) AS DuplicatedQuestionsCount
FROM (
        SELECT Duplicated.RelatedPostId, Posts.OwnerUserId
        FROM (
                SELECT PostLinks.RelatedPostId
                FROM PostLinks
                WHERE PostLinks.LinkTypeId == 3
             ) AS Duplicated
        JOIN Posts
        ON Duplicated.RelatedPostId = Posts.Id
     ) AS DuplicatedPosts
JOIN Users ON Users.Id == DuplicatedPosts.OwnerUserId
GROUP BY Users.Id
HAVING DuplicatedQuestionsCount > 100
ORDER BY DuplicatedQuestionsCount DESC
```

```sql
--- 5)
SELECT QuestionsAnswers.Id,
       QuestionsAnswers.Title,
       QuestionsAnswers.Score,
       MAX(Duplicated.Score) AS MaxScoreDuplicated,
       COUNT(*) AS DulicatesCount,
       CASE
         WHEN QuestionsAnswers.Hour < '06' THEN 'Night'
         WHEN QuestionsAnswers.Hour < '12' THEN 'Morning'
         WHEN QuestionsAnswers.Hour < '18' THEN 'Day'
         ELSE 'Evening'
         END DayTime
FROM (
        SELECT Id, Title,
               STRFTIME('%H', CreationDate) AS Hour, Score
        FROM Posts
        WHERE Posts.PostTypeId IN (1, 2)
     ) AS QuestionsAnswers
JOIN (
        SELECT PL3.RelatedPostId, Posts.Score
        FROM (
               SELECT RelatedPostId, PostId
               FROM PostLinks
               WHERE LinkTypeId == 3
             ) AS PL3
        JOIN Posts ON PL3.PostId = Posts.Id
     ) AS Duplicated
ON QuestionsAnswers.Id = Duplicated.RelatedPostId
GROUP BY QuestionsAnswers.Id
ORDER By DulicatesCount DESC
```