.

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 http://www.gagolewski.com/resources/data/travel_stackexchange_com/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 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. 3 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 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
```

```
--- 2)
SELECT
    Users.DisplayName,
    Users.Age,
    Users.Location,
    SUM(Posts.FavoriteCount) AS FavoriteTotal,
    Posts.Title AS MostFavoriteQuestion,
    MAX(Posts.FavoriteCount) AS MostFavoriteQuestionLikes
FROM Posts
JOIN Users ON Users.Id=Posts.OwnerUserId
WHERE Posts.PostTypeId=1
GROUP BY OwnerUserId
ORDER BY FavoriteTotal DESC
LIMIT 10
```

```
--- 3)
SELECT
    Posts.Title,
    CmtTotScr.CommentsTotalScore
FROM (
        SELECT
            PostID,
            UserID,
            SUM(Score) AS CommentsTotalScore
        FROM Comments
        GROUP BY PostID, UserID
) AS CmtTotScr
JOIN Posts ON Posts.ID=CmtTotScr.PostID AND Posts.OwnerUserId=CmtTotScr.UserID
WHERE Posts.PostTypeId=1
ORDER BY CmtTotScr.CommentsTotalScore DESC
LIMIT 10
```