Basic SQL queries

Deadline: 23:59 10.3.2024 for all parts of the assignment (AIS, Github and reporting the link to the Docker image via the form). The assignment is worth a maximum of 7 points.

Overview

The task is aimed at creating basic SQL queries over the attached PostgreSQL database, which was created from the Stack Exchange Data Dump Superuser dataset.

The goal is to implement the below tasks as RESTful endpoints, which are implemented as SQL queries, transformed into JSON output. The output is described as a JSON Schema.

The inputs to connect to the database server will be provided in the same way as in Assignment 1 (using environment variables). You can develop the assignment as a extension. Just add the handler for HTTP endpoints described below.

The order in the output must be the same as its definition at each end-point. Only plain SQL queries may be used in the implementation and no ORMs are allowed. For responses, the times must be returned in ISO8601 format in UTC.

In addition to implementing the endpoints themselves, documentation needs to be produced, which will include:

- SQL queries with description,
- examples of HTTP endpoint calls (for each endpoint).

The documentation can be implemented as PDF or Markdown documentation with the understanding that it will reside in the AIS commit and also in the GitHub repository itself.

HTTP Endpoints

GET /v2/posts/:post_id/users

Return a list of all discussants (users) of the post (posts) with the ID *:post_id*, sorting them according to when their comment was made, starting with the newest and ending with the oldest.

The JSON schema of the HTTP response is in the *schemas/users.json* file. An example response for post ID 1819157 can be found in the 1 block.


```
{
  "items": [
      {
        "id": 1866388,
        "reputation": 1,
        "creationdate": "2023-12-01T00:05:24.3+00:00",
        "displayname": "TomR.",
        "lastaccessdate": "2023-12-03T06:18:19.607+00:00",
        "websiteurl": null,
        "location": null,
        "aboutme": null,
        "views": 1,
        "upvotes": 0,
        "downvotes": 0,
        "profileimageurl": null,
        "age": null,
        "accountid": 30035903
      }
  ]
}
```


Listing 1: GET /v2/posts/1819157/users

GET /v2/users/:user_id/friends

Produce a discussion list for the user *user_id*, containing users who have commented on posts that the user has created or commented on. Organize the users according to their registration date, starting with those who registered first.

The JSON schema of the HTTP response is in the *schemas/users.json* file. An example response for user ID 1076348 is found in the 2 block.

```
{
  "items": [
  {
    "id": 482362,
    "reputation": 10581,
    "creationdate": "2015-08-11T17:42:36.267+02",
    "displayname": "DrZoo",
    "lastaccessdate": "2023-12-03T06:41:11.75+01",
    "websiteurl": null,
    "location": null,
    "aboutme": null,
    "views": 1442,
    "upvotes": 555,
    "downvotes": 46,
    "profileimageurl": null,
    "age": null,
    "accountid": 2968677
  },
  {
    "id": 1076348,
    "reputation": 1,
    "creationdate": "2019-08-15T16:00:28.473+02",
    "displayname": "Richard",
    "lastaccessdate": "2019-09-10T16:57:48.527+02",
    "websiteurl": null,
    "location": null,
    "aboutme": null,
    "views": 0,
    "upvotes": 0,
    "downvotes": 0,
    "profileimageurl": null,
    "age": null,
    "accountid": 16514661
  }
]
}
```


Listing 2: GET /v2/users/1076348/users

GET /v2/tags/:tagname/stats

Determine the percentage of posts with a particular tag within the total number of posts published on each day of the week (e.g. Monday, Tuesday), for each day of the week separately. Show the results on a scale of 0 - 100 and round to two decimal places.

The JSON schema of the HTTP response is in the *schemas/posts-load.json* file. An example response for the *linux* tag can be found in the 3 block.

```
{
  "result": {
    "monday": 11.53,
    "tuesday": 11.62,
    "wednesday": 11.52,
    "thursday": 11.35,
    "friday": 11.68,
    "saturday": 12.04,
    "sunday": 11.82
  }
}
```

Listing 3: GET /v2/tags/linux/stats

GET /v2/posts/?duration=:duration_in_minutes&limit=:limit

The output is a list of the *:limit* of the most recently resolved posts that have been opened for a maximum of *:duration_in_minutes* minutes (the number of minutes between *creationdate* and *closeddate*). Round the opening duration (*duration*) to two decimal places.

The JSON schema of the HTTP response is in the *schemas/posts.json* file. An example response for limit *2* and duration *5* is found in the 4 block.


```
{
  "items": [
    {
      "id": 1818849,
      "creationdate": "2023-11-30T16:55:32.137+01",
      "viewcount": 22924,
      "lasteditdate": null,
      "lastactivitydate": "2023-11-30T16:55:32.137+01",
      "title": "Why is my home router address is 10.x.x.x and not 100.x.x.x which is
       ↪  properly reserved and widely accepted for CGNAT?",
      "closeddate": "2023-11-30T16:59:23.56+01",
      "duration": 3.86
    },
    {
      "id": 1818386,
      "creationdate": "2023-11-27T18:26:57.617+01",
      "viewcount": 19,
      "lasteditdate": null,
      "lastactivitydate": "2023-11-27T18:26:57.617+01",
      "title": "Are there any libraries for parsing DWG files with LGPL, MIT, Apache, BSD?",
      "closeddate": "2023-11-27T18:29:18.947+01",
      "duration": 2.36
    }
  ]
}
```


Listing 4: GET /v2/posts?duration=5&limit=2

GET /v2/posts?limit=:limit&query=:query

Suggest an endpoint that provides a list of posts ordered from newest to oldest. Include a complete list of associated tags as part of the response. This endpoint supports two parameters:

- *limit*: the maximum number of posts in a reply,
- *query*: search string over *posts.title* and *posts.body*.

The search is not case sensitive.

The JSON schema of the HTTP response is in the *schemas/post-search.json* file. An example response for query *linux* and limit *1* can be found in block 5.


```
{
  "items": [
    {
      "id": 1819160,
      "creationdate": "2023-12-03 05:22:43.587+01",
      "viewcount": 7,
      "lasteditdate": null,
      "lastactivitydate": "2023-12-03 05:22:43.587+01",
      "title": "Keyboard not working on khali linux",
      "body": "<p>I have recently installed virtualbox on my windows 10 and trying to run
       ↪  Linux Ubuntu and Kali. Everything working on Ubuntu without any issue but when I
       ↪  am running kali it is not taking keyboard(Samsung bluetooth 500) input. Please can
       ↪  anyone help me out here.\nMany thanks in advance!!</p>\n",
      "answercount": 0,
      "closeddate": null,
      "tags": [
        "virtual-machine"
      ]
    }
  ]
}
```


Listing 5: GET /v2/posts?limit=:limit&query=:query

Instructions for submission

Each assignment must be published as a Github Release. To automatically create a docker image, you need to use a CI/CD script that will automatically create the image.

Your final release for an assignment needs to be marked according to the semantic versioning, so for assignment 2 it is marked as 2.x.x. For successful submission of the assignment, all the points listed below must be met:

- The source code (so that the project can be compiled and run) together with the commit hash (which represents the final version of your assignment 2) has been uploaded to **AIS**.
- On **GitHub**, a private repository is maintained within the GitHub Classroom.
- The address of your Docker Image has been reported using **Google Form** https://forms.gle/tReJgMwMVrJYBLcj8

An assignment is considered complete if all parts (AIS, Github, Link to docker image) are submitted by the commit deadline. In case of late submission, the assignment is considered as not submitted.

The assignment will be scored based on a table 1.

The assignment is presented in front of the lector. If the student is unable to defend his/her solution, his/her assignment may not be accepted and will be marked zero.

Description	**Points**
Dokumentácia	2
GET /v2/posts/:post_id/users	1
GET /v2/users/:user_id/friends	1
GET /v2/tags/:tagname/stats	1
GET /v2/posts/?duration=:duration_in_minutes&limit=:limit	1
GET /v2/posts?limit=:limit&query=:query	1

Table 1: Evaluation