

# Social Computing Homework Coursebook

## Instructions

Please fill in each exercise and submit the entire document as a PDF on Moodle before the section's respective deadline. Keep working on the whole document so that for the last submission you submit a completely filled in template. You may not change previous sections in subsequent submissions. Some sections require you to work on an existing software project, which you have to fork on [GitHub.com](https://github.com), or clone and create your repository. Provide the URL of your public fork or repository of this project below.

Fill in each answer to a homework task to the textbox underneath. Use as much space as you wish. Do not provide long code snippets or other irrelevant information.

## Restrictions

You may use AI tools for language styling or only. Usage of any AI tools to answer questions, inspire creative solutions or write code is strictly forbidden. Group work and sharing solutions is strictly prohibited. Any suspected cases of [misconduct](#) will be referred to the Education Dean. If you are not sure whether you are in violation of course-specific restrictions or the university's code of conduct, please ask the Lecturer or a TA.

### Name

*Mohammadhosein Mohammadirad*

### Student ID

*2409311*

### Student Email

*mmohamma24@student.oulu.fi*

### GitHub Repository URL

*<https://github.com/mhmohammadirad/SocialComputing>*

## AI Use Disclaimer

**Explain in detail in what parts and how AI was used for any of the work above. Fill it out and update after each homework submission, even if you did not use AI at all.**

**Your answers to homework tasks should not include AI-generated code or text.**

*No use of AI yet*

## Task 1 (due 22.9.2025 23:59)

15 points

**Exercise 1.1** Reading the dataset: Load the database and for each table, print and inspect the available columns and the number of rows. Explain below how you loaded the database. For each table, describe all columns (name, purpose, type, example of contents). You may use SQL and/or Python to perform this task. (3 points)

*I used SQLite (by Alexcvzz) and SQLite viewer (by Florian Klampfer) extensions in VSCode to load and view the database. I know that using sqlite3 library in python is also an approach, but I used this extension and it is much easier to work with SQLite database. It loads the database so you can see the tables, the table definitions (DDL), and the column data types, and even data in each table.*

*There are 6 tables in the database. I will go through each of these tables, their columns, their purposes, and data types.*

*comments table (5 columns, 5804 rows): This table contains information about comments, including which user has created this comment for which post and when.*

- *id: unique identifier ranging from 1 to 5804 with INTEGER type.*
- *post\_id: foreign key to posts table with INTEGER type to show which post this comment is related to.*
- *user\_id: foreign key to users table with INTEGER type to show which user has created this comment*
- *content: the content of the comment with TEXT type e. g.: Don't miss this! Everyone is talking about it ? viralbuzz.news*
- *created\_at: creation date and time of the comment with TIMESTAMP type. It is in YYYY-MM-DD hh:mm:ss format.*

*follows table (2 columns, 7225 rows): This table contains information about following relationships, showing which user follows which other user*

- *follower\_id: foreign key to users table with INTEGER type to show which user is following someone (follower)*
- *followed\_id: foreign key to users table with INTEGER type to show which user is being followed by someone (following)*

*posts table (4 columns, 1303 rows): This table contains information about posts published on this platform, including the content of the post, its creator, and time of creation.*

- *id: unique identifier ranging from 1718 to 3020 with INTEGER type.*
- *user\_id: foreign key to users table with INTEGER type to show which user has created this post*
- *content: the content of the post with TEXT type e. g.: Revolutionary idea! #fashionblogger #instafashion #model #runway #hairstyle #ootd #trend*
- *created\_at: creation date and time of the post with TIMESTAMP type. It is in YYYY-MM-DD hh:mm:ss format.*

*reactions table (4 columns, 8276 rows): This table contains information about reactions to the posts, including which user has reacted to which post, how was the reaction, and when.*

- *id: unique identifier ranging from 1 to 8286 with INTEGER type.*
- *post\_id: foreign key to posts table with INTEGER type to show which post this reaction is related to.*
- *user\_id: foreign key to users table with INTEGER type to show which user has reacted to the post*
- *reaction\_type: reaction to that post with TEXT type, but it is somehow an ENUM because it has only 6 different values: like, wow, love, haha, sad, angry. I used this query to find them sorted based on their frequency:*

```
SELECT reaction_type, COUNT(1) AS count
FROM reactions
GROUP BY reaction_type
ORDER BY 2 DESC;
```

*sqlite\_sequence (2 columns, 3 rows): This table contains the sequences used for unique identifiers in our table. There are three entries in this table, each of them showing the last used id in a sequence (for the related table). SQLite uses these sequences for autoincrement id sequences.*

- *name: name of the sequence with UNKNOWN type which is the name of the table this sequence is related to e. g. reactions, comments, posts*
- *seq: last used number for that sequence.*

*users table (7 columns, 210 rows): This table contains information about users on the platform, including their username, password, location, and birthdate.*

- *id: unique identifier ranging from 1 to 533 with INTEGER type.*
- *username: username of the user with VARCHAR(50) type. E. g. traveller\_tom*
- *location: location (city) of the user with VARCHAR(100) type. E. g. Berlin, Germany*
- *birthdate: birthdate of the user with DATE type. E. g. 1990-10-12*
- *created\_at: date and time that the user account was created with TIMESTAMP type. E. g. 2022-05-17 17:32:48*
- *profile: some info that users mention about themselves (like bio) with TEXT type. E. g. Berlin native, born on 10/12/90. ? Cinephile with a penchant for thought-provoking films and ...*
- *password: password of the user with TEXT type. E. g. wLx6NQRI*

**Exercise 1.2** Lurkers: How many users are there on the platform who have not interacted with posts or posted any content yet (but may have followed other users)? Answer and explain your queries/calculations below. You may use SQL and/or Python to perform this task. (3 points)

*There are 55 users who have never interacted with posts or created any content.*

*If we replace COUNT(1) with \* we can see the details of these users.*

*WHERE clause uses three subqueries:*

- *All user ids who have created at least one post from posts table.*
- *All user ids who have reacted to at least one post from reactions table.*
- *All user ids who have written at least one comment (this one is optional, since the exercise is asking for posts and reaction. But posts already cover them, so it doesn't affect the result).*

*The main query then selects all users whose IDs are not in any of these subqueries.*

```
SELECT COUNT(1) AS count
FROM users
WHERE id NOT IN (SELECT DISTINCT user_id FROM posts)
  AND id NOT IN (SELECT DISTINCT user_id FROM reactions)
  AND id NOT IN (SELECT DISTINCT user_id FROM comments);
```

**Exercise 1.3 Influencers:** In the history of the platform, who are the 5 users with the most engagement on their posts? Describe how you measure engagement. Answer and explain your queries/calculations below. You may use SQL and/or Python to perform this task. (4 points)

*For calculating reactions for each user, first we need to join reactions table with posts table to find the publisher of each post and find how many reactions they have received for each of their posts. We do so in the subquery reacts, so we will know which user, on which post, has received how many reactions.*

*Then, to find the users with highest reaction, we can have two approaches:*

- 1) If we consider the count of all reactions each user have received (sum of all reactions of their posts) then we have these users as the top 5 users with most reactions: 54, 94, 65, 81, 111*
- 2) but as you can see, these users can have a lot of posts with a few reactions on each of them which at the end, creates a high number of overall received reactions, but in fact, they don't receive much reactions, they just post a lot. So we can have a better approach, average reactions that each user receive for each of their posts. In that case, these users have the highest engagement: 78, 43, 109, 94, 13*

*To calculate engagement based on reactions, we first need to know how many reactions each user has received on their posts. To do this, we join the reactions table with the posts table in order to connect each reaction to the publisher of the post. In the subquery reacts, we calculate how many reactions each post has received, while keeping track of which user owns that post. This gives us a dataset showing, for each user and each post, the number of reactions received.*

*From this point, there are two possible approaches to identify the top 5 users with the most engagement:*

- *Total reactions approach: We sum up all the reactions that each user has received across all their posts. This way, we find the users who have the largest total number of reactions. Using this method, the top 5 users are: 54, 94, 65, 81, 111.*

- *Average reactions approach: A potential problem with the first method is that a user could have many posts with only a few reactions each. Their total reactions would still be high, but it wouldn't necessarily mean their individual posts are highly engaging. To account for this, we can instead look at the average number of reactions per post for each user. This shows us who consistently receives strong engagement on their posts, regardless of how many they make. Using this method, the top 5 users are: 78, 43, 109, 94, 13.*

*The query below calculates both total reactions and average reactions per post. By changing the ORDER BY clause, we can switch between the two approaches:*

```
WITH reacts AS (
    SELECT posts.user_id, post_id, COUNT(1) AS count
    FROM reactions JOIN posts ON reactions.post_id = posts.id
    GROUP BY posts.user_id, post_id
)
SELECT user_id, SUM(count) AS total_reactions,
       COUNT(1) AS total_posts, AVG(count) AS avg_reactions_per_post
FROM reacts
GROUP BY user_id
-- ORDER BY 2 DESC -- Using total_reactions(approach 1)
ORDER BY 4 DESC -- Using avg_reactions_per_post (approach 2)
LIMIT 5;
```

**Exercise 1.4 Spammers:** Identify users who have shared the same text in posts or comments at least 3 times over and over again (in all their history, not just the last 3 contributions). Answer and explain your queries/calculations below. You may use SQL and/or Python to perform this task. (5 points)

*There are 5 users who have posted or commented the same text at least 3 times.  
ids: 513, 521, 524, 530, 533*

*First, we combine all posts and comments into one dataset (all\_data).  
Then, we group by user\_id and content to find cases where the same user repeated the same text 3 or more times (spams).  
Finally, we select the ids of those users.*

```
WITH all_data AS (
    SELECT user_id, content FROM posts
    UNION ALL
    SELECT user_id, content FROM comments
), spams AS (
    SELECT user_id, content, COUNT(1) AS count
    FROM all_data
    GROUP BY user_id, content
    HAVING COUNT(1) >= 3
```

```
)  
SELECT DISTINCT user_id  
FROM spams;
```

## Task 2 (due 29.9.2025 23:59)

15 points

**Exercise 2.1** Growth: This year, we are renting 16 servers to run our social media platform. They are soon at 100% capacity, so we need to rent more servers. We would like to rent enough to last for 3 more years without upgrades, plus 20% capacity for redundancy. We need an estimate of how many servers we need to start renting based on past growth trends. Plot the trend on a graph using Python and include it below. Answer and explain your queries/calculations below. You may use SQL and/or Python to perform this task. (Note that the dataset may not end in the current year, please assume that the last data marks today's date) (3 points)

*Write your answer here...*

**Exercise 2.2** Virality: Identify the 3 most viral posts in the history of the platform. Select and justify a specific metric or requirements for a post to be considered viral. Answer and explain your queries/calculations below. You may use SQL and/or Python to perform this task. (4 points)

*Write your answer here...*

**Exercise 2.3** Content Lifecycle: What is the average time between the publishing of a post and the first engagement it receives? What is the average time between the publishing of a post and the last engagement it receives? Answer and explain your queries/calculations below. You may use SQL and/or Python to perform this task. (4 points)

*Write your answer here...*

**Exercise 2.4** Connections: Identify the top 3 user pairs who engage with each other's content the most. Define and describe your metric for engagement. Answer and explain your queries/calculations below. You may use SQL and/or Python to perform this task. (4 points)

*Write your answer here...*



### Task 3 (due 19.10.2025 23:59)

15 points

**Exercise 3.1** Censorship: implement the `moderate_content` function that automatically detects and censors inappropriate user posts on the platform. Your function should take a post, comment or user introduction as input and apply censorship rules to either clean or remove content, and supply a risk score that corresponds to the number and weight of violations in the content (note the risk classification thresholds in the code). The exact rules are detailed on the Rules page. Think of and implement one more moderation measure you think is important to keep the platform safe. Include and explain your implementation below. (5 points)

*Write your answer here...*

**Exercise 3.2** User risk analysis: Assign risk scores to each user by implementing the `user_risk_analysis` function. This function returns a risk score for a given user based on rules presented on the Rules page. Identify the top 5 highest risk users. Think of and implement one more risk prediction measure you think is important to keep the platform safe. Answer and explain your queries/calculations below. (5 points)

*Write your answer here...*

**Exercise 3.3** Recommendation Algorithm: Implement the `recommend` function. Identify a suitable, simple recommendation algorithm that will recommend 5 relevant posts on the “Recommended” tab based on the posts the user reacted to positively and the users they followed. (5 points)

*Write your answer here...*

## Task 4 (due 27.10.2025 23:59)

20 points

**Exercise 4.1** Topics: Identify the 10 most popular topics discussed on our platform. Use Latent Dirichlet Allocation (LDA) with the `gensim` library. Answer and explain your queries/calculations below. You may use SQL and/or Python to perform this task. (5 points)

*Write your answer here...*

**Exercise 4.2** Sentiment: Perform sentiment analysis on posts and comments. What is the overall tone of the platform? How does sentiment vary across user posts discussing different topics identified in Exercise 3? Please use VADER (`nltk.sentiment`) for this analysis. Answer and explain your queries/calculations below. You may use SQL and/or Python to perform this task. (5 points)

*Write your answer here...*

**Exercise 4.3** Learning from others' mistakes: Find two social platforms similar to Mini Social that have been under fire for an engineering, design or operation error that severely affected a large group of users. Describe how we can learn from their mistakes and draft up a plan about how Mini Social can be improved learning from their mistakes. You do not need to write code in this exercise unless your plan includes a specific change to an algorithm or function. (5 points)

*Write your answer here...*

**Exercise 4.4** Design and implement a new social feature in Mini Social. For example, a user reputation scoring system, a reporting system, a feature to find related content to a post, new post modalities such as polls or reposts. Your change must include a UI improvement or addition. Do not implement non-social, technical features, such as resource optimization, security improvements or style changes. Document the design and implementation process of your addition here. You must also demonstrate a fully functional feature in a maximum 2-minute video recording uploaded to Moodle. (5 points)

*Write your answer here...*

