Assignment 2: SQL INDIVIDUAL PROJECT (10%)

Name: Vinh Nguyen (ID 1029531)

/\*

1. List the full names (e.g. Alice Smith), as one column, of the users who have not taken any Steps yet. (1)

\*/

SELECT CONCAT(first\_name, ' ', last\_name) AS full\_name

FROM user

WHERE id NOT IN (SELECT DISTINCT user\_id FROM step\_taken);



8 row(s) returned

/\*

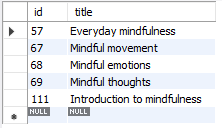
2. List all of the IDs and titles of Steps that contain the substring ‘mindful’ in their title. (1)

\*/

SELECT id, title

FROM step

WHERE title LIKE '%mindful%';



5 row(s) returned

/\*

3. Provide a list of the titles of all Steps completed by user with id = 17. Do not show duplicates (list each title only once). (2)

\*/

SELECT DISTINCT title

FROM

# Step IDs completed by user with id = 17

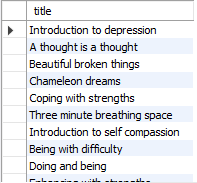
(SELECT DISTINCT step\_id

FROM step\_taken

WHERE user\_id = 17 AND when\_finished IS NOT NULL) AS step\_user17

LEFT JOIN step

ON step\_user17.step\_id = step.id;



16 row(s) returned

/\*

4. Provide a list of the titles of all Steps that have been taken more than two times along with a count of how many times. (2)

\*/

SELECT step.title, popular\_step.step\_count

FROM

# Steps that have been taken more than two times

(SELECT step\_id, COUNT(id) AS step\_count

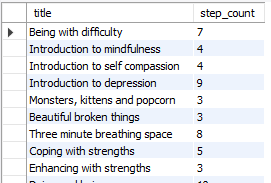
FROM step\_taken

GROUP BY step\_id

HAVING step\_count > 2) AS popular\_step

LEFT JOIN step

ON popular\_step.step\_id = step.id;



11 row(s) returned

/\*

5. Which Step(s), listed with columns id, title and the count of times taken, have been taken the greatest number of times? (3)

\*/

SELECT most\_popular\_step.step\_id, step.title, most\_popular\_step.step\_count

FROM

# Step IDs taken the greatest number of times

(SELECT step\_id, COUNT(id) AS step\_count

FROM step\_taken

GROUP BY step\_id

HAVING step\_count =

# Max count

(SELECT COUNT(id) AS step\_count

FROM step\_taken

GROUP BY step\_id

ORDER BY step\_count DESC

LIMIT 1)

) AS most\_popular\_step

LEFT JOIN step

ON most\_popular\_step.step\_id = step.id;



1 row(s) returned

/\*

6. List each Step with the title column, along with a count of how many times that Step has been taken and the average

rating received by the Step (formatted to 2 decimal places). Order the result by the average rating as a number in

descending order. (3)

\*/

SELECT step.title, average\_rating.step\_count, average\_rating.avg\_rating

FROM

# how many times a Step has been taken and the average rating received by the Step

(SELECT step\_id, COUNT(id) AS step\_count, FORMAT(AVG(rating), 2) AS avg\_rating

FROM step\_taken

GROUP BY step\_id) AS average\_rating

RIGHT JOIN step

ON average\_rating.step\_id = step.id

ORDER BY average\_rating.avg\_rating DESC;



49 row(s) returned

/\*

7. Provide a list of the titles of all Steps that have been taken by both Alice (id = 1) and Bob (id == 2), along with the

combined number of times they have taken the Step. (4)

\*/

SELECT step.title, user\_1\_2.step\_count

FROM

# Step IDs and times taken by both user id 1 and user id 2

(SELECT step\_id, COUNT(id) AS step\_count

FROM

# Steps taken by either user id 1 or user id 2

(SELECT \*

FROM step\_taken

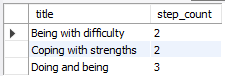
WHERE user\_id IN (1, 2)) AS user\_1\_2

GROUP BY step\_id

HAVING COUNT(DISTINCT user\_id) = 2) AS user\_1\_2

LEFT JOIN step

ON user\_1\_2.step\_id = step.id;



3 row(s) returned

/\*

8. List users older than or equal to 21 years of age, along with a count of how many other users they are following

and a count of how many other users are following them. List the user’s id, first name, last name, age, following

count and followed count, and order the results by first name ascending, then last name ascending. (4)

\*/

SELECT follow\_21.\*, COUNT(user\_follow.following\_user\_id) AS followed\_count

FROM

# Following count included

(SELECT follow\_21.\*, COUNT(user\_follow.followed\_user\_id) as following\_count

FROM

# Age >= 21

(SELECT id, first\_name, last\_name, TIMESTAMPDIFF(YEAR, DOB, CURRENT\_TIMESTAMP) AS age

FROM user

WHERE TIMESTAMPDIFF(YEAR, DOB, CURRENT\_TIMESTAMP) >= 21) AS follow\_21

LEFT JOIN user\_follow

ON follow\_21.id = user\_follow.following\_user\_id

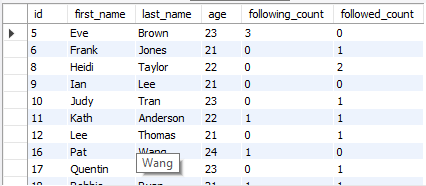
GROUP BY follow\_21.id) AS follow\_21

LEFT JOIN user\_follow

ON follow\_21.id = user\_follow.followed\_user\_id

GROUP BY follow\_21.id

ORDER BY follow\_21.first\_name ASC, follow\_21.last\_name ASC;



10 row(s) returned

/\*

9. For each (user, theme) pair such that user has taken some steps under the theme, provide a count of how many times

a user has taken a step that is categorised under the theme. The output should consist of user ID, user first name,

user last name, theme name and the count of steps taken. (5)

\*/

SELECT user\_theme.user\_id, user.first\_name, user.last\_name, theme.name AS theme\_name, user\_theme.step\_count

FROM

# user\_id|theme\_id|step\_count

(SELECT step\_taken.user\_id, step\_theme.theme\_id, COUNT(step\_taken.id) AS step\_count

FROM

step\_taken

INNER JOIN step\_theme

ON step\_taken.step\_id = step\_theme.step\_id

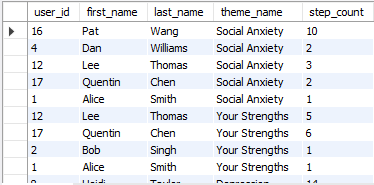
GROUP BY step\_taken.user\_id, step\_theme.theme\_id) AS user\_theme

INNER JOIN user

ON user\_theme.user\_id = user.id

INNER JOIN theme

ON user\_theme.theme\_id = theme.id;



42 row(s) returned

/\*

10. A) Provide a complete list of all user ID pairs such that the two users follow each other and share at least one

interest. (Hint: MySQL has a CROSS JOIN operator, which returns the Cartesian product of rows from the joined

tables) (5)

\*/

SELECT pair.\*

FROM

# Users follow each other

(SELECT uf1.following\_user\_id AS user\_id1, uf1.followed\_user\_id AS user\_id2

FROM

user\_follow AS uf1

CROSS JOIN user\_follow AS uf2

WHERE

uf1.following\_user\_id < uf1.followed\_user\_id

AND uf2.following\_user\_id > uf2.followed\_user\_id

AND (uf1.following\_user\_id, uf1.followed\_user\_id) = (uf2.followed\_user\_id, uf2.following\_user\_id)

) AS pair

INNER JOIN user\_interest AS ui1

ON pair.user\_id1 = ui1.user\_id

INNER JOIN user\_interest AS ui2

ON pair.user\_id2 = ui2.user\_id

GROUP BY pair.user\_id1, pair.user\_id2

HAVING SUM(ui1.interest\_id = ui2.interest\_id) >= 1;



1 row(s) returned