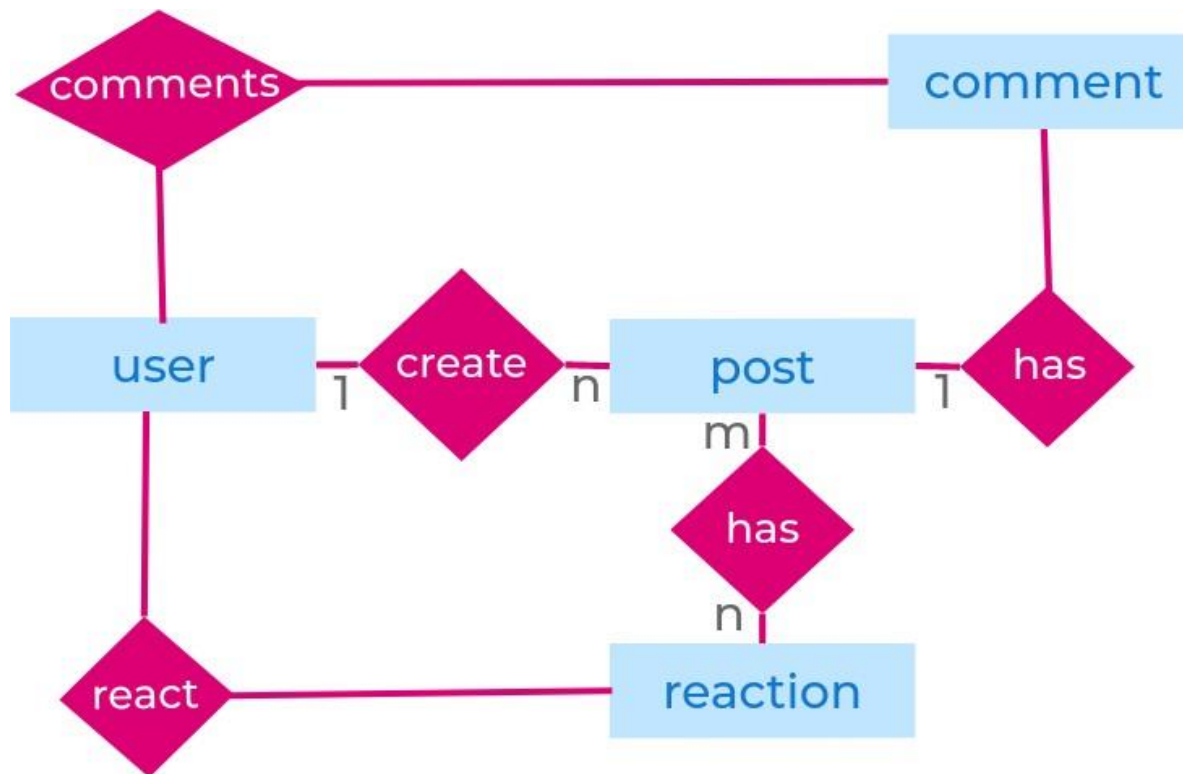# Coding Practice - 03

In this practice set, let's apply Joins to fetch meaningful data from a sample database of social networking application.

Database:

The database contains user, post, comment and reaction tables.

* 1:n relation between user and
* comment 1:1 relation between
* user and reaction 1:n relation
  between post and comment

Note

You can assume that a user may or may not post on the social media platform, i.e., a user can have zero or many posts. Also, a post can have zero or many comments or reactions. So, do take care of this detail while writing various queries on the database, especially, when deciding on INNER JOIN / LEFT JOIN.

Refer the tables in the code playground for a better understanding of the database.

User Table

user_id	name	age	gender
1	Danielle Adkins	15	F
2	Timothy Becker	27	M

Reaction table

reaction_id	reaction_type	reacted_by	post_id	reacted_at
0	LIKE	62	29	2021-02-14 02:27:32
1	DISLIKE	85	20	2021-02-11 11:25:54

Comment Table

comment_id	comment_content	commented_by	post_id	commented_at
0	Group beat want movement course	65	23	2021-01-01 11:09:32
1	Growth activity	1	18	2021-01-01 11:09:32
2	Focus structure mention	32	6	2021-01-01 11:09:32

Post Table

post_id	content	posted_by	posted_at
1	Keep like small	58	2021-01-01 11:09:32
2	Military blue	22	2021-01-01 11:09:32

QUESTIONS

1. Fetch all the posts along with user details.

   Note:

   Sort the posts in the descending order of

posted_at

Expected Output Format:

name	age	post_id	content	posted_at
...	...	...	...	...

SELECT user.name, user.age, post.post_id, post.content, post.posted_at FROM user INNER JOIN post ON user.user_id = post.posted_by ORDER BY posted_at DESC;

2. Fetch the 5 recent posts posted by "James Williams". ✓

   Note:

   - Condiser that the name of the user is "James Williams"
   - Get the recent posts based on posted_at column in post table

   Expected Output Format:

name	age	post_id	content	posted_at
...	...	...	...	...

SELECT user.name, user.age, post.post_id, post.content, post.posted_at FROM user INNER JOIN post ON user.user_id = post.posted_by WHERE user.name = 'James Williams' ORDER BY posted_at DESC LIMIT 5;

3. Fetch the total number of posts posted by "James Williams" till date as ✓

   posts_count

   Note:

   - Condiser that the name of the user is "James Williams"

   Expected Output Format:

posts_count
...

SELECT count(post.post_id) AS posts_count FROM user INNER JOIN post ON user.user_id = post.posted_by WHERE user.name = 'James Williams';

4. For every user, fetch the total number of posts posted so far as ✓

   posts_count

   Note:

   - If a user does not publish any posts, keep the posts_count as 0.

- Sort the output in the descending order of posts_count, and then in the ascending order of user_id.

user_id	user_name	posts_count
...	...	...

SELECT user.user_id, user.name AS user_name, count(post.post_id) AS posts_count FROM user LEFT JOIN post ON user.user_id = post.posted_by GROUP BY user_name, user.user_id ORDER BY posts_count DESC, user.user_id ASC;

5. Get all the users details who posted at least 2 posts. ✅

   Note:

   - Sort the output in the descending order of posts_count, and then in the ascending order of user_id.

   Expected Output Format:

user_id	user_name	age	gender	posts_count
...	...	...	...	...

SELECT user.user_id, user.name AS user_name, user.age, user.gender, count(post.post_id) AS posts_count FROM user LEFT JOIN post ON user.user_id = post.posted_by GROUP BY user_name, user.user_id HAVING posts_count >= 2 ORDER BY posts_count DESC, user.user_id ASC;

6. Fetch the active users in January 2021. ✅

   Note:

   - Consider the users as active, if they posted at least 2 posts in January 2021.
   - Sort the output in the descending order of posts_count, and then in the ascending order of user_id.

   Expected Output Format:

user_id	user_name	posts_count
...	...	...

SELECT user.user_id, user.name AS user_name, count(post.post_id) AS posts_count FROM user LEFT JOIN post ON user.user_id = post.posted_by WHERE strftime("%Y", posted_at) = '2021' AND strftime("%m", posted_at) = '01' GROUP BY user_name, user.user_id HAVING posts_count >= 2 ORDER BY posts_count DESC, user.user_id ASC;

7. For each post, get the total number of positive reactions as ✅

positive_reactions_count

   Note:

   - Consider the reaction_type "LIKE", "LIT", "WOW" and "LOVE" as positive reactions.
   - If the post does not have any positive reactions, keep thepositive_reactions_count as 0.
   - Sort the output in the descending order of positive_reactions_count , and then in the ascending order of post_id .

   Expected Output Format:

post_id	content	posted_by	positive_reactions_count
...	...	...	...

SELECT post.post_id, post.content, post.posted_by, count(CASE WHEN reaction.reaction_type IN ('LIKE', 'LIT', 'WOW', 'LOVE') THEN 1 ELSE NULL END) AS positive_reactions_count FROM post LEFT JOIN reaction ON reaction.post_id = post.post_id GROUP BY post.post_id ORDER BY positive_reactions_count DESC, post.post_id;

8. For each post, get the total number of positive reactions and negative reactions as ✅

positive_reactions_count and negative_reactions_count respectively.

Note:

- Consider the reaction_type "LIKE", "LIT", "WOW" and "LOVE" as positive reactions, and "DISLIKE" as a negative reaction.
- If the post does not have positive_reactions, keep the positive_reactions_count as 0.
- If the post does not have any negative_reactions, keep the negative_reactions_count as 0.
- Sort the output in the ascending order of post_id.

Expected Output Format:

post_id	content	positive_reactions_count	negative_reactions_count
...	...	...	...

SELECT post.post_id, post.content, count(CASE WHEN reaction.reaction_type IN ('LIKE', 'LIT', 'WOW', 'LOVE') THEN 1 ELSE NULL END) AS positive_reactions_count, count(CASE WHEN reaction.reaction_type IN ('DISLIKE') THEN 1 ELSE NULL END) AS negative_reactions_count FROM post LEFT JOIN reaction ON reaction.post_id = post.post_id GROUP BY post.post_id ORDER BY post.post_id;

9. A general manager from the social networking site would like to review the content in all the posts that have ✅ more negative reactions over positive reactions.

Note:

- Consider the reaction_type "LIKE", "LIT", "WOW" and "LOVE" as positive reactions, and "DISLIKE" as a negative reaction.
- Output must contain rows in the ascending order of post_id.

Expected Output Format:

post_id	content	positive_reactions_count	negative_reactions_count
....	....	....	....

SELECT post.post_id, post.content, count(CASE WHEN reaction.reaction_type IN ('LIKE', 'LIT', 'WOW', 'LOVE') THEN 1 ELSE NULL END) AS positive_reactions_count, count(CASE WHEN reaction.reaction_type IN ('DISLIKE') THEN 1 ELSE NULL END) AS negative_reactions_count FROM post INNER JOIN reaction ON reaction.post_id = post.post_id GROUP BY post.post_id HAVING positive_reactions_count < negative_reactions_count ORDER BY post.post_id;

10. Get all the comment details for the post with post_id = 5. ✅

   Note:

   - Output must contain comments in the ascending order of commented_at.

   Expected Output Format:

comment_id	comment_content	commented_by	commented_at
...	...	...	...

SELECT comment_id, comment_content, commented_by, commented_at FROM COMMENT WHERE post_id = 5 ORDER BY commented_at;

11. Fetch all the posts posted by the user with user_id = 98, along with the comments for the posts. ✅

   Note:

   - Output must contain posts in the ascending order of post_id, and then in the descending order of commented_at.
   - If there is no comment for a post, keep the comment_id and comment_content as NULL.

   Expected Output Format:

post_id	posted_by	content	posted_at	comment_id	comment

SELECT t.post_id AS post_id, t.posted_by AS posted_by, t.content AS content, t.posted_at AS posted_at, t.comment_id AS comment_id, t.comment_content AS comment_content, t.commented_at AS commented_at FROM (post LEFT JOIN COMMENT ON COMMENT.post_id = post.post_id) AS T LEFT JOIN user ON user.user_id = t.posted_by WHERE user_id = 98 ORDER BY post.post_id ASC, COMMENT.commented_at DESC;

12. Fetch all the posts and their comments count, for the posts posted by the user with user_id = 98. ✅

   Note:

   - Output must contain posts in the ascending order of post_id.
   - If there is no post for a user, keep the content, posted_at values as NULL.
   - If there is no comment for a post, keep the comments_count values as 0.

   Expected Output Format:

post_id	posted_by	content	posted_at	comments_count
...	...	...	...	...

SELECT t.post_id AS post_id, t.posted_by AS posted_by, t.content AS content, t.posted_at AS posted_at, count(t.comment_id) AS comments_count FROM (post LEFT JOIN COMMENT ON COMMENT.post_id = post.post_id) AS T LEFT JOIN user ON user.user_id = t.posted_by WHERE user_id = 98 GROUP BY post.post_id ORDER BY post.post_id ASC;

13. For each user, get all the posts posted, and the total comments count for each post, along with user_details. ✅

    Note:

    - Sort the output in the ascending order of user_id, and then in the ascending order of post_id.
    - If there is no post for a user, keep the content, posted_at values as NULL
    - If there is no comment for a post, keep the comments_count value as 0.

    Expected Output Format:

user_id	name	post_id	content	posted_at	comment
...	...	...	...	...	...

SELECT user_id AS user_id, t.name AS name, t.post_id AS post_id, t.content AS content, t.posted_at AS posted_at, count(COMMENT.comment_id) AS comments_count FROM (user LEFT JOIN post ON user.user_id = post.posted_by) AS T LEFT JOIN COMMENT ON COMMENT.post_id = t.post_id GROUP BY post.post_id, user.user_id ORDER BY user.user_id ASC, post.post_id ASC;

14. For each user, get all the posts posted, and the total reactions count for each post, along with user_details.

    Note:

    Sort the output in the ascending order of user_id, and then in the ascending order of post_id.If there is no post for a user, keep the content, posted_at as NULL.

    If there is no reaction for a post, keep the reactions_count as 0.

    Expected Output Format:

user_id	name	post_id	content	posted_at	Reactions_count
...	...	...	...	...	...

SELECT t.user_id AS user_id, t.name AS name, t.post_id AS post_id, t.content AS content, t.posted_at AS posted_at, count(reaction.reaction_id) AS reactions_count FROM (user LEFT JOIN post ON user.user_id = post.posted_by) AS T LEFT JOIN reaction ON reaction.post_id = t.post_id GROUP BY post.post_id, user.user_id ORDER BY user.user_id ASC, post.post_id ASC;