# **Problem 1 (Meta Hard Level)**

A table named “famous” has two columns called user id and follower id. It represents each user ID has a particular follower ID. These follower IDs are also users of #Facebook / Meta. Then, find the famous percentage of each user.

Famous Percentage = number of followers a user has / total number of users on the platform.

**Explanation:**

1. distinct\_users CTE: Combines user\_id and follower\_id using UNION to get all unique users on the platform. This helps us determine the total number of users.
2. follower\_count CTE: Counts the number of followers for each user\_id by grouping the rows in the famous table. This gives a list of users with their follower counts.
3. Final SELECT Statement: Uses the data from follower\_count and distinct\_users to calculate the famous percentage for each user.

**MySQL Solution:**



**MSSQL Solution:**



# **Problem 2 (Amazon Hard Level)**

Given a table 'sf\_transactions' of purchases by date, calculate the month-over-month percentage change in revenue. The output should include the year-month date (YYYY-MM) and percentage change, rounded to the 2nd decimal point, and sorted from the beginning of the year to the end of the year. The percentage change column will be populated from the 2nd month forward and calculated as

.

**Explanation:**

1. MonthlyRevenue CTE: Aggregates the total revenue for each month using FORMAT to convert the created\_at date to the format YYYY-MM.
2. RevenueChange CTE: Adds a column previous\_revenue using the LAG function, which fetches the total revenue of the previous month for each row.
3. Final SELECT: Calculates the percentage change as ((total\_revenue - previous\_revenue) / previous\_revenue) \* 100. The ROUND function ensures the percentage is rounded to two decimal places. The output is ordered by year\_month to display the data chronologically.

**MSSQL Server Solution:**

****

**MySQL Solution:**

****

# **Problem 3 (Google Medium Level)**

You are analyzing a social network dataset at Google. Your task is to find mutual friends between two users, Karl and Hans. There is only one user named Karl and one named Hans in the dataset.  
  
The output should contain 'user\_id' and 'user\_name' columns.

**Explanation:**

1. The CTEs (karl\_friends and hans\_friends) efficiently find all friends for Karl and Hans, respectively.
2. The main query joins these CTEs with the users table to find the users who are present in both Karl's and Hans's friend lists (mutual friends).

