# **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).



# **Problem 4 (Uber Hard Level)**

Some forecasting methods are extremely simple and surprisingly effective. Naïve forecast is one of them. To create a naïve forecast for "distance per dollar" (defined as distance\_to\_travel / monetary\_cost), first sum the "distance to travel" and "monetary cost" values monthly. This gives the actual value for the current month. For the forecasted value, use the previous month's value. After obtaining both actual and forecasted values, calculate the root mean squared error (RMSE) using the formula

Report the RMSE rounded to two decimal places.



**MySQL Solution:**



**MSSQL Solution:**



# **Problem 5 (Microsoft Medium Level)**

Given a list of projects and employees mapped to each project, calculate by the amount of project budget allocated to each employee. The output should include the project title and the project budget rounded to the closest integer. Order your list by projects with the highest budget per employee first.

Explanation:

1. Joining Tables: The initial step involves joining the ms\_projects and ms\_emp\_projects tables on the project ID to combine project details (including titles and budgets) with employee assignments.
2. Grouping and Aggregating: The data is then grouped by project title and budget, allowing for the calculation of budget per employee by dividing the total budget of each project by the count of employees assigned to that project.
3. Rounding and Ordering: Finally, the computed budget per employee is rounded to the nearest integer, and the results are ordered in descending order to prioritize projects with the highest budget allocation per employee.



# **Problem 6 (Airbnb Medium Level)**

Find the total number of available beds per hosts' nationality. Output the nationality along with the corresponding total number of available beds. Sort records by the total available beds in descending order.

**Explanation**

1. Joining Tables: The first step involves joining the airbnb\_apartments and airbnb\_hosts tables on the host\_id. This allows us to combine the apartment details (such as the number of beds) with the host's nationality information.
2. Grouping and Aggregating: Next, the data is grouped by the host's nationality, so that the total number of beds available for each nationality can be calculated. The SUM() function is used to add up the beds (n\_beds) for all apartments hosted by individuals of the same nationality.
3. Sorting the Results: Finally, the results are ordered in descending order based on the total number of available beds.



# **Problem 7 (IBM Hard Level)**

IBM is working on a new feature to analyze user purchasing behavior for all Fridays in the first quarter of the year. For each Friday separately, calculate the average amount users have spent per order. The output should contain the week number of that Friday and average amount spent.

𝐓𝐨 𝐬𝐨𝐥𝐯𝐞 𝐭𝐡𝐢𝐬 𝐩𝐫𝐨𝐛𝐥𝐞𝐦, 𝐰𝐞 𝐧𝐞𝐞𝐝 𝐭𝐨:  
1. Identify the Fridays in the first quarter (Q1) of the year.  
2. Calculate the week number for each of these Fridays.  
3. Group the purchases by week number and calculate the average amount spent per order.

**MSSQL Solution**



**MySQL Solution:**



# **Problem 8 (Tesla Medium Level)**

You are given a table of product launches by company by year. Write a query to count the net difference between the number of products companies launched in 2020 with the number of products companies launched in the previous year. Output the name of the companies and a net difference of net products released for 2020 compared to the previous year.

**Explanation:**

1. Counting Products per Year: Using SUM with CASE statements, we count the number of products launched in 2020 and 2019 separately for each company.
2. Calculating Net Difference: We calculate the difference between 2020 and 2019 product counts to get the net change.
3. Ordering: The results are ordered by net\_difference in descending order to show companies with the highest increase first.



# **Problem 9 (Netflix Hard Problem)**

Find the genre of the person with the most number of oscar winnings.  
If there are more than one person with the same number of oscar wins, return the first one in alphabetic order based on their name. Use the names as keys when joining the tables.

**Explanation:**

1. WinnerCount CTE: Calculates the total Oscar wins for each nominee by counting rows where winner = 1.
2. Final Selection: The TOP 1 clause fetches all rows with the highest total\_wins, sorted alphabetically by name to handle ties. We join the WinnerCount CTE with nominee\_information on the nominee’s name to retrieve the top\_genre for the top nominee(s) in terms of Oscar wins.

**MSSQL Solution**



**MySQL Solution**



# **Problem 10 (Amazon Medium Level)**

Write a query that'll identify returning active users. A returning active user is a user that has made a second purchase within 7 days of any other of their purchases. Output a list of user\_ids of these returning active users.

**MSSQL Solution**

****

**MySQL Solution**

****

# **Problem 11 (Nvidia, Microsoft Medium Level)**

Find the number of transactions that occurred for each product. Output the product name along with the corresponding number of transactions and order records by the product id in ascending order. You can ignore products without transactions.

**Explanation:**

1. Joining Tables: The INNER JOIN between excel\_sql\_inventory\_data (aliased as inv) and excel\_sql\_transaction\_data (aliased as trans) matches records by product\_id. This way, only products with transactions are included.
2. Counting Transactions: Using COUNT(trans.transaction\_id) counts the number of transactions for each product.
3. Grouping and Ordering: GROUP BY inv.product\_id, inv.product\_name groups by product\_id and product\_name to get the transaction count per product. ORDER BY inv.product\_id ASC sorts the output by product\_id in ascending order.



# **Problem 12 (LinkedIn, Dropbox Basic Level)**

Write a query that calculates the difference between the highest salaries found in the marketing and engineering departments. Output just the absolute difference in salaries.

**Explanation**

1. CASE is used to selectively get the salary for the "marketing" and "engineering" departments.
2. MAX is applied to retrieve the highest salary in each department.
3. ABS calculates the absolute difference between the two maximum values.

