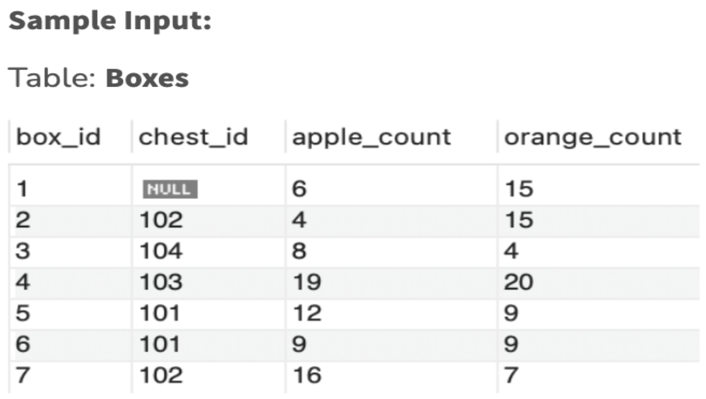
Apples and Oranges

https://www.scaler.com/hire/test/problem/40114/

Problem Description:

Write a query to count the number of apples and oranges in all the boxes. If a box contains a chest, you should also include the number of apples and oranges it has.



Sample Input:

Table: **Boxes**

box_id	chest_id	apple_count	orange_count
1	NULL	6	15
2	102	4	15
3	104	8	4
4	103	19	20
5	101	12	9
6	101	9	9
7	102	16	7

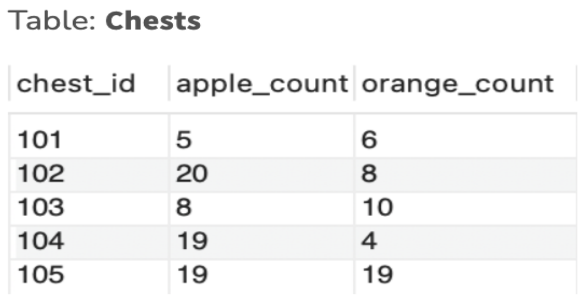
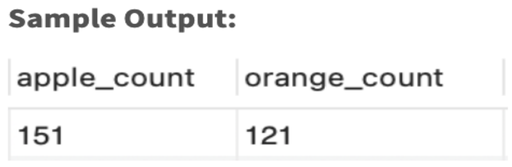


Table: **Chests**

chest_id	apple_count	orange_count
101	5	6
102	20	8
103	8	10
104	19	4
105	19	19



Sample Output:

apple_count	orange_count
151	121

Sample Explanation:

- box 1 has 6 apples and 15 oranges.
- box 2 has 4 + 20 (from the chest) = 24 apples and 15 +8 (from the chest) = 23 oranges.
- box 3 has 8 + 19 (from the chest) = 27 apples and 4 + 4 (from the chest) = 8 oranges.
- box 4 has 19 + 8 (from the chest) = 27 apples and 20 + 10 (from the chest) = 30 oranges.
- box 5 has 12 + 5 (from the chest) = 17 apples and 9 + 6 (from the chest) = 15 oranges.

- box 6 has 9 + 5 (from the chest) = 14 apples and 9 + 6 (from the chest) = 15 oranges.
- box 7 has 16 + 20 (from the chest) = 36 apples and 7 + 8 (from the chest) = 15 oranges.
- Total number of apples = 6 + 24 + 27 + 27 + 17 + 14 + 36 = 151.
- Total number of oranges = 15 + 23 + 8 + 30 + 15 + 15 + 15 = 121

Unpopular Books

https://www.scaler.com/hire/test/problem/41750/

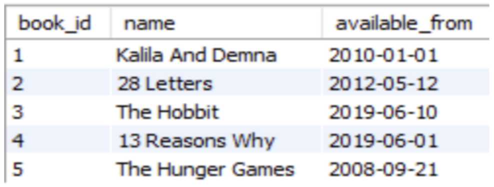
Problem Statement:

Write a query that reports the books that have sold less than 10 copies in the last year, excluding books that have been available for less than one month from today. Assume today is 2019-06-23.

- Return the result table ordered by book_id.

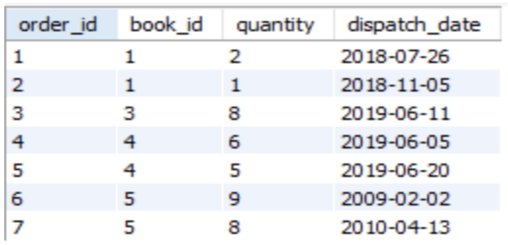
Sample Input:

Table: books



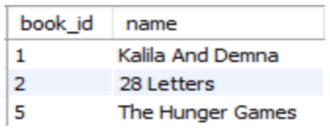
book_id	name	available_from
1	Kalila And Demna	2010-01-01
2	28 Letters	2012-05-12
3	The Hobbit	2019-06-10
4	13 Reasons Why	2019-06-01
5	The Hunger Games	2008-09-21

Table: orders



order_id	book_id	quantity	dispatch_date
1	1	2	2018-07-26
2	1	1	2018-11-05
3	3	8	2019-06-11
4	4	6	2019-06-05
5	4	5	2019-06-20
6	5	9	2009-02-02
7	5	8	2010-04-13

Sample output:



book_id	name
1	Kalila And Demna
2	28 Letters
5	The Hunger Games

Explanation:

- It can be clearly observed that 'Kalila And Demna' sold less than 10 copies in the last year. Also, it is available for more than one month from today.
- A similar case is with the books '28 Letters' and 'The Hunger Games'.

Ads Performance

https://www.scaler.com/hire/test/problem/41760/

Problem Description:

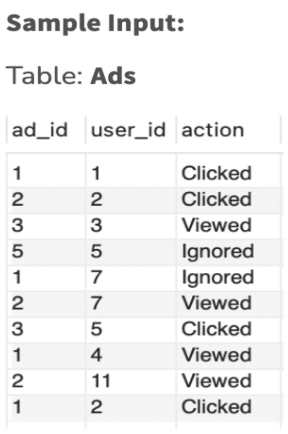
A company is running Ads and wants to calculate the performance of each Ad.

Performance of the Ad is measured using Click-Through Rate (CTR) where:

$$CTR = \begin{cases} 0, & \text{if Ad total clicks + Ad total views} = 0 \\ \frac{\text{Ad total clicks}}{\text{Ad total clicks + Ad total views}} \times 100, & \text{otherwise} \end{cases}$$

Write a query to find the CTR of each Ad.

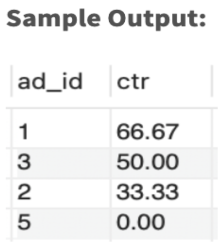
- Round ctr to two decimal points.
- Return the output ordered by ctr in descending order and by ad_id in ascending order in case of a tie.



Sample Input:

Table: **Ads**

ad_id	user_id	action
1	1	Clicked
2	2	Clicked
3	3	Viewed
5	5	Ignored
1	7	Ignored
2	7	Viewed
3	5	Clicked
1	4	Viewed
2	11	Viewed
1	2	Clicked



Sample Output:

ad_id	ctr
1	66.67
3	50.00
2	33.33
5	0.00

Sample Explanation:

- for ad_id = 1, ctr = (2/(2+1)) * 100 = 66.67
- for ad_id = 2, ctr = (1/(1+2)) * 100 = 33.33
- for ad_id = 3, ctr = (1/(1+1)) * 100 = 50.00
- for ad_id = 5, ctr = 0.00, Note that ad_id = 5 has no clicks or views.
- Note that we do not care about Ignored Ads.

Page Recommendations

https://www.scaler.com/hire/test/problem/41763/

Problem Description:

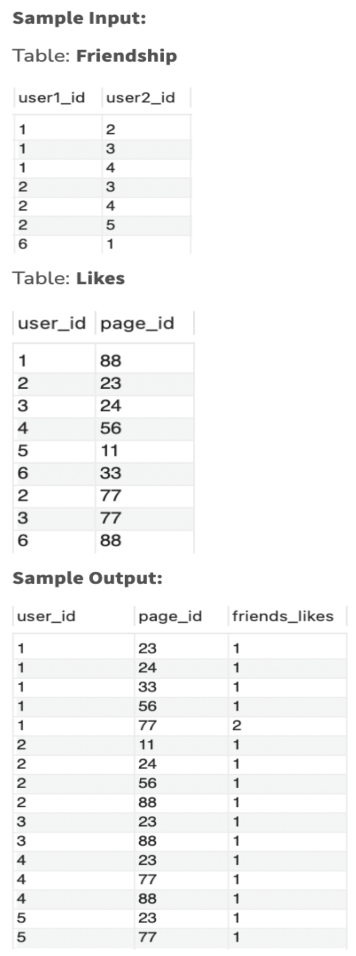
You are implementing a page recommendation system for a social media website. Your system will recommend a page to user_id if the page is liked by at least one friend of user_id and is not liked by user_id.

Write a query to find all the possible page recommendations for every user.

Each recommendation should appear as a row in the output table with these columns:

- user_id: The ID of the user that your system is making the recommendation to.
- page_id: The ID of the page that will be recommended to user_id.
- friends_likes: The number of the friends of user_id that like page_id.

- Return the output ordered by user_id and page_id in ascending order.



Sample Input:

Table: **Friendship**

user1_id	user2_id
1	2
1	3
1	4
2	3
2	4
2	5
6	1

Table: **Likes**

user_id	page_id
1	88
2	23
3	24
4	56
5	11
6	33
2	77
3	77
6	88

Sample Output:

user_id	page_id	friends_likes
1	23	1
1	24	1
1	33	1
1	56	1
1	77	2
2	11	1
2	24	1
2	56	1
2	88	1
3	23	1
3	88	1
4	23	1
4	77	1
4	88	1
5	23	1
5	77	1

Sample Explanation:

Take user 1 as an example:

- User 1 is friends with users 2, 3, 4, and 6.
- Recommended pages are 23 (user 2 liked it), 24 (user 3 liked it), 56 (user 3 liked it), 33 (user 6 liked it), and 77 (user 2 and user 3 liked it).
- Note that page 88 is not recommended because user 1 already liked it.

Another example is user 6:

- User 6 is friends with user 1.
- User 1 only liked page 88, but user 6 already liked it.
- Hence, user 6 has no recommendations.

You can recommend pages for users 2, 3, 4, and 5 using a similar process.

Strictly Increasing Purchases

https://www.scaler.com/hire/test/problem/41764/

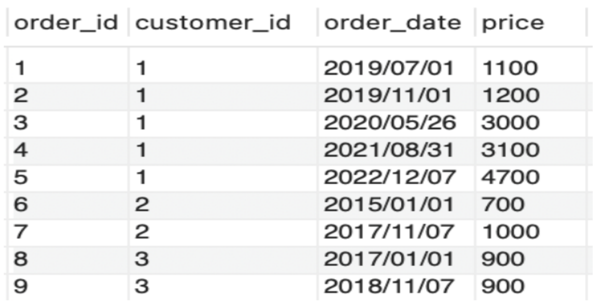
Problem Description:

Write a query to report the IDs of the customers with the total purchases strictly increasing yearly.

- The total purchases of a customer in one year is the sum of the prices of their orders in that year.
- If for some year the customer did not make any order, we consider the total purchases 0.
- The first year to consider for each customer is the year of their first order.
- The last year to consider for each customer is the year of their last order.
- Return the output ordered by customer_id in ascending order.

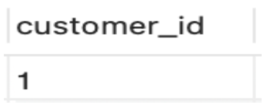
Sample Input:

Table: **Orders**



order_id	customer_id	order_date	price
1	1	2019/07/01	1100
2	1	2019/11/01	1200
3	1	2020/05/26	3000
4	1	2021/08/31	3100
5	1	2022/12/07	4700
6	2	2015/01/01	700
7	2	2017/11/07	1000
8	3	2017/01/01	900
9	3	2018/11/07	900

Sample Output:



customer_id
1

Sample Explanation:

Customer 1: The first year is 2019 and the last year is 2022

- 2019: 1100 + 1200 = 2300
- 2020: 3000
- 2021: 3100
- 2022: 4700
- We can see that the total purchases are strictly increasing yearly, so we include customer 1 in the answer.

Customer 2: The first year is 2015 and the last year is 2017

- 2015: 700
- 2016: 0
- 2017: 1000
- We do not include customer 2 in the answer because the total purchases are not strictly increasing. Note that customer 2 did not make any purchases in 2016.

Customer 3: The first year is 2017, and the last year is 2018

- 2017: 900
- 2018: 900
- We can see that the total purchases are not strictly increasing yearly they are the same in both years.

Hence, we include only the customer with customer_id 1 in the output.

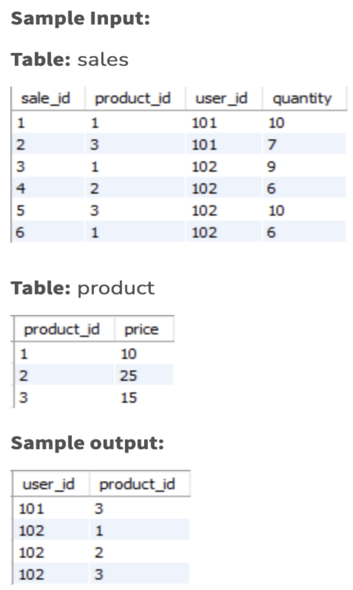
Product Sales Analysis IV

https://www.scaler.com/hire/test/problem/41877/

Problem Statement:

Write a query that reports for each user the product id on which the user spent the most money. In case the same user spent the most money on two or more products, report all of them.

- Return the resulting table ordered by user_id in an ascending manner.



Sample Input:

Table: sales

sale_id	product_id	user_id	quantity
1	1	101	10
2	3	101	7
3	1	102	9
4	2	102	6
5	3	102	10
6	1	102	6

Table: product

product_id	price
1	10
2	25
3	15

Sample output:

user_id	product_id
101	3
102	1
102	2
102	3

Explanation:

- User 101:
  - Spent 10 * 10 = 100 on product 1.
  - Spent 7 * 15 = 105 on product 3.
- User 101 spent the most money on product 3.
- User 102:
  - Spent (9 + 7) * 10 = 150 on product 1.
  - Spent 6 * 25 = 150 on product 2.
  - Spent 10 * 15 = 150 on product 3.
- User 102 spent the most money on products 1, 2, and 3.

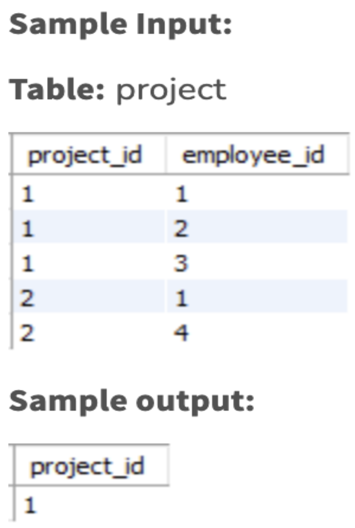
Project Employees II

https://www.scaler.com/hire/test/problem/41720/

Problem Statement:

Write a query that reports all the projects that have the most employees.

- Return the result table ordered by the project_id column.



Sample Input:

Table: project

project_id	employee_id
1	1
1	2
1	3
2	1
2	4

Sample output:

project_id
1

Explanation:

The first project has 3 employees while the second one has 2.

Solution:

STEP 1: In the outer query, select the column project_id from the project table.

STEP 2: Using the group-by-clause, group the data based on the project_id, and in the having clause use the count() aggregate function and count the number of employees for each project.

STEP 3: In the inner query select the employee_id and using the count() function count the employees for each project.

- Use the order by clause and order the count of employees in descending order.
- Use the limit clause and retrieve the highest count of the employee.

STEP 4: Now, from the outer query filter the records that match the count from the inner query.

STEP 5: Use the order by clause and order the data in ascending order based on the project_id.

Answers:
https://docs.google.com/document/d/15DpqW2Jfw61Yahow-AN14FAqsiftkG9j4EhcMahyEO0/edit

DML to insert records:

https://docs.google.com/document/d/1EBO0JUCQtgCt4nLvqPcCJc22pw5CvQHG6ciL8YjYpGo/edit?usp=sharing