資料庫管理 HW03

B12508026 戴偉璿

1. (a) Left join all advisors(e) and their advisees(s), if someone has no advisee, then s would be NULL.

```sql
select e.id, e.name from employee as e
left join employee s on e.id=s.supervisor_id
where s.supervisor_id is null;
```

(b) Find the latest store id of each employee before 2025-01-05 and left join to the employee table.

```sql
select e.id as employee_id, h.store_id
from employee e
left join employee_store_history h
on e.id = h.employee_id
and h.start_date_time=(
    select max(h2.start_date_time)
    from employee_store_history h2
    where h2.employee_id = e.id
        and h2.start_date_time <= '2025-01-05'
    );
```

(c) Using `limit 1` to obtain the first store id and `limit 1 offset 1` to obtain the second store id (after skipping the first one), then join them to produce the final result.

```sql
select e.id as employee_id,
    (select h1.store_id from employee_store_history h1 where h1.employee_id=e.id order by
    ↪ h1.start_date_time limit 1) as first_store_id,
    (select h2.store_id from employee_store_history h2 where h2.employee_id=e.id order by
    ↪ h2.start_date_time limit 1 offset 1) as second_store_id
from employee e;
```

(d) Calculating the total quantity purchased for each product, then ordering by total quantity and product_id, using `limit 2 offset 3` to find the 4th and 5th products. Finally, joining with purchase_detail and purchase tables to get the required information.

```sql
with total_qty as(
    select pd.product_id as product_id, sum(pd.qty) as total_qty, count(*) as purchase_count
    from purchase_detail pd
```

```
4        group by pd.product_id
5    ),
6    target_product as(
7        select product_id
8        from total_qty
9        order by total_qty desc, product_id asc
10       limit 2 offset 3
11   )
12   select p.id as product_id, p.name as product_name, pu.store_id as store_id, count(*) as
     ↪  purchase_count, sum(pd.qty) as total_qty
13   from target_product tp
14   join product p on tp.product_id = p.id
15   join purchase_detail pd on pd.product_id = p.id
16   join purchase pu on pd.purchase_no = pu.purchase_no
17   group by p.id, p.name, pu.store_id
18   order by p.id, pu.store_id;
```

(e) First choose the target product ids by ranking the total purchased quantity, then cross join with store table to get all combinations of target products and stores. Finally, left join with purchase and purchase_detail tables to get the required information.

```
1    with total_qty as(
2        select pd.product_id as product_id, row_number() over(order by sum(pd.qty) desc,
     ↪  pd.product_id asc) as rnk
3        from purchase_detail pd
4        group by pd.product_id
5    ),
6    target as(
7        select product_id
8        from total_qty
9        where rnk >= 4 and rnk <= 5
10   )
11   select t.product_id as id, p.name as name, s.id as store_id, coalesce(sum(pd.qty), 0) as amount,
     ↪  coalesce(count(distinct pu.purchase_no), 0) as cnt
12   from target t --target store id
13   cross join store s
14   left join purchase pu on pu.store_id = s.id
15   left join purchase_detail pd on pu.purchase_no = pd.purchase_no and t.product_id = pd.product_id
16   join product p on p.id = t.product_id
17   group by s.id, t.product_id, p.name
18   order by t.product_id, s.id;
```

(f) Calculate the total spending of each member in each store, then rank them within each store based on the total spending. Using `rank()` instead of `row_number()` to handle ties in spending amounts. While ranking, using `partition by` to separate rankings for each store.

```
1    select sa.store_id as store_id, sa.member_id as member_id, sum(sd.unit_price*sd.qty) as amount,
     ↪  rank() over(partition by sa.store_id order by sum(sd.unit_price*sd.qty) desc) as rnk
2
3    from sales sa
4    join sales_detail sd on sd.receipt_no = sa.receipt_no
5    where member_id is not null
```

```
6  group by sa.store_id, sa.member_id
7  order by sa.store_id, rnk;
```

2. (a) First, I identified the hard questions based on the number of total answers and correct answers. Then, for each hard question, I found each user's first correct submission and ranked the users by cost time and submission timestamp. Finally, I counted the number of hard questions where each user ranked in the top 3 and kept only those who appeared in at least two questions.

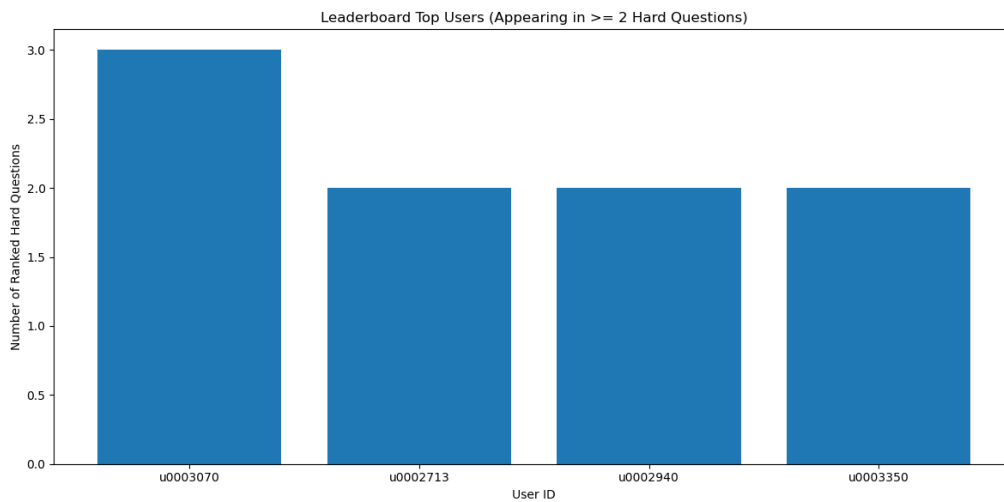Fig. 1 shows the leaderboard of top users appearing in at least two hard questions.

Figure 1: Leaderboard of Top Users Appearing in at Least Two Hard Questions

Following is my source code, I used DuckDB to execute the SQL query and Matplotlib to visualize the results.

```
1  import duckdb
2  import matplotlib.pyplot as plt
3
4  con = duckdb.connect('my_database.db')
5
6  df = con.sql("""
7  -- list the hard questions
8  with hard_questions as(
9      select question_id as id, count(*) as total_ans, sum(is_correct) as total_ac
10     from answers
11     group by question_id
12     having total_ans >= 1000 and total_ac <= 500
13 ),-- list the rank for all users
14 first_correct_pre as(
15     select question_id, user_id, cost_time, created_at, rank() over(partition by question_id,
        ↪  user_id order by created_at asc) as rnk
16     from answers
17     where is_correct = 1
18 ),-- first correct for each user
19 first_correct as(
20     select question_id, user_id, cost_time, created_at
```

```
21      from first_correct_pre
22      where rnk = 1
23  ),--rank users for hard questions
24  usr_rnk as(
25      select question_id, user_id, rank() over(partition by question_id order by cost_time asc,
     ↪   created_at asc) as rnk
26      from first_correct fc
27      join hard_questions hq on hq.id = fc.question_id
28  )
29  select user_id, count(distinct question_id) as appr_cnt
30  from usr_rnk
31  where rnk <= 3
32  group by user_id
33  having appr_cnt >= 2
34  order by appr_cnt desc, user_id asc;
35  """).df()
36  print(df)
37
38  plt.figure(figsize=(12,6))
39  plt.bar(df["user_id"].astype(str), df["appr_cnt"])
40  plt.xlabel("User ID")
41  plt.ylabel("Number of Ranked Hard Questions")
42  plt.title("Leaderboard Top Users (Appearing in >= 2 Hard Questions)")
43  plt.tight_layout()
44  plt.savefig("2a_leaderboard.png")
45  plt.show()
```

(b) Using the same CTEs as in 2a to find the users who ranked in the top 3 for hard questions. Then I calculated their scores based on the given scoring system. Finally, I kept only those users with a total score of at least 5.

Fig. 2 shows the leaderboard of top users with a score of at least 5.
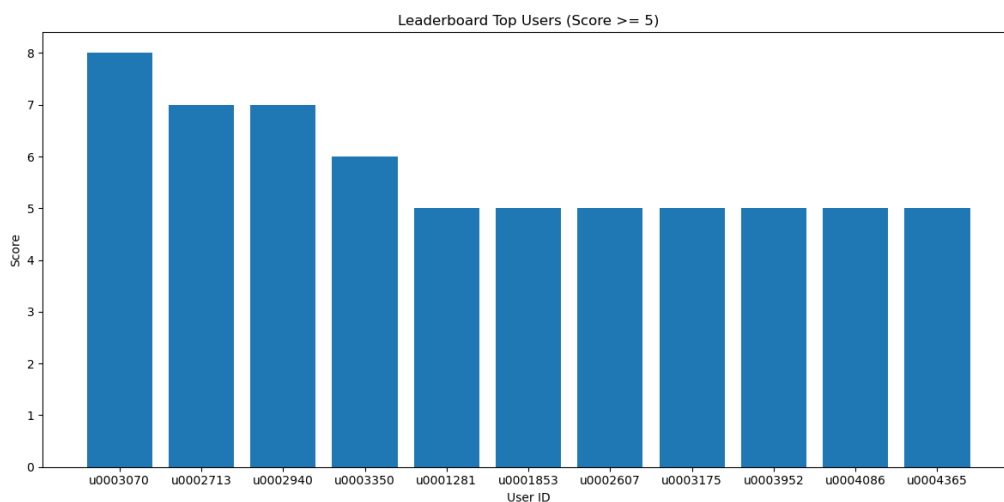

Figure 2: Leaderboard of Top Users with a Score of at Least 5

Following is my source code:

```
1  import duckdb
2  import matplotlib.pyplot as plt
```

```python
3
4   con = duckdb.connect('my_database.db')
5
6   df = con.sql("""
7   -- list the hard questions
8   with hard_questions as(
9       select question_id as id, count(*) as total_ans, sum(is_correct) as total_ac
10      from answers
11      group by question_id
12      having total_ans >= 1000 and total_ac <= 500
13  ),-- list the rank for all users
14  first_correct_pre as(
15      select question_id, user_id, cost_time, created_at, rank() over(partition by question_id,
        ↪  user_id order by created_at asc) as rnk
16      from answers
17      where is_correct = 1
18  ),-- first correct for each user
19  first_correct as(
20      select question_id, user_id, cost_time, created_at
21      from first_correct_pre
22      where rnk = 1
23  ),--rank users for hard questions
24  usr_rnk as(
25      select question_id, user_id, rank() over(partition by question_id order by cost_time asc,
        ↪  created_at asc) as rnk
26      from first_correct fc
27      join hard_questions hq on hq.id = fc.question_id
28  )
29  select user_id,
30      sum(
31          cast(rnk = 1 as int) * 5 +
32          cast(rnk = 2 as int) * 2 +
33          cast(rnk = 3 as int) * 1
34      ) as score
35  from usr_rnk
36  where rnk <= 3
37  group by user_id
38  having score >= 5
39  order by score desc, user_id asc;
40  """).df()
41  print(df)
42
43  plt.figure(figsize=(12,6))
44  plt.bar(df["user_id"].astype(str), df["score"])
45  plt.xlabel("User ID")
46  plt.ylabel("Score")
47  plt.title("Leaderboard Top Users (Score >= 5)")
48  plt.tight_layout()
49  plt.savefig("2b_leaderboard.png")
50  plt.show()
```