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
```

```
    group by pd.product_id
),
target_product as(
    select product_id
    from total_qty
    order by total_qty desc, product_id asc
    limit 2 offset 3
)
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
from target_product tp
join product p on tp.product_id = p.id
join purchase_detail pd on pd.product_id = p.id
join purchase pu on pd.purchase_no = pu.purchase_no
group by p.id, p.name, pu.store_id
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.

```
with total_qty as(
    select pd.product_id as product_id, row_number() over(order by sum(pd.qty) desc,
    ↪ pd.product_id asc) as rnk
    from purchase_detail pd
    group by pd.product_id
),
target as(
    select product_id
    from total_qty
    where rnk >= 4 and rnk <= 5
)
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
from target t --target store id
cross join store s
left join purchase pu on pu.store_id = s.id
left join purchase_detail pd on pu.purchase_no = pd.purchase_no and t.product_id = pd.product_id
join product p on p.id = t.product_id
group by s.id, t.product_id, p.name
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.

```
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

from sales sa
join sales_detail sd on sd.receipt_no = sa.receipt_no
where member_id is not null
```

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