力扣高频 SQL 50 题阶段总结（五）

> 本组题目属于 LeetCode 高频 SQL 中的进阶部分，覆盖了 **排名、窗口函数、关系网络、交换行、TopN 查询、UNION 语义陷阱** 等经典考点。

一、1978 – 员工的直接上司（层级关系）

https://leetcode.cn/problems/employees-whose-manager-left-the-company/description/?envType=study-plan-v2&envId=sql-free-50

题目描述

给定员工表 Employees：

- employee_id：员工编号
- name：员工姓名
- manager_id：该员工直属上司的 employee_id（可能为空）

要求输出每个员工的直接上司信息。

本题考察点

- **层级关系在 SQL 中如何表示**
- manager_id 指向同一张表，是典型的“自引用外键”

解题思路（拆解）

1. 表中每一行是一个员工
2. manager_id 并不是上司名字，而是上司的 employee_id
3. 因此需要再次在 Employees 表中找到：
    - employee_id = manager_id 的那一行
4. 使用 self join（自连接）完成匹配
5. 对于 manager_id 为空的员工，需要保留，因此用 LEFT JOIN

示例解法

代码块

```
SELECT e.employee_id,
```

```
        e.name AS employee_name,
        m.name AS manager_name
FROM Employees e
LEFT JOIN Employees m
ON e.manager_id = m.employee_id;
```

易错点

- 使用 INNER JOIN 会丢失没有上司的员工
- 忘记区分员工别名 e 和经理别名 m

二、626 – 换座位（行交换）

https://leetcode.cn/problems/exchange-seats/?envType=study-plan-v2&envId=sql-free-50

题目描述

Seat 表按 id 排序表示座位顺序，要求：

- 两两交换相邻学生
- 若最后一个学生没有相邻座位，则保持不变

核心考点

> **行号奇偶性 + CASE 条件映射**

解题思路（详细拆解）

1. id 为奇数的学生，需要和后一个交换 → id + 1
2. id 为偶数的学生，需要和前一个交换 → id - 1
3. 但如果总人数为奇数，则最后一个奇数 id 没有后继，不能 +1
4. 因此需要额外判断：当前 id 是否是最大 id

示例解法

代码块

```
SELECT
    CASE
        WHEN id % 2 = 1 AND id != (SELECT MAX(id) FROM Seat)
            THEN id + 1
        WHEN id % 2 = 0
            THEN id - 1
        ELSE id
```

```
    END AS id,
    student
FROM Seat
ORDER BY id;
```

易错点

- 忘记处理最后一个奇数 id
- CASE 顺序写反导致逻辑错误

三、1341 – 电影评分（UNION ALL 陷阱）

https://leetcode.cn/problems/movie-rating/?envType=study-plan-v2&envId=sql-free-50

题目描述

要求输出两行结果：

1. 评分次数最多的用户（若并列取名字最小）
2. 2020 年 2 月平均评分最高的电影（若并列取标题最小）

本题核心

- 两个独立查询，各自 LIMIT 1
- 最后拼接成两行输出

解题思路（详细拆解）

第一问：评分次数最多的用户

1. MovieRating 表中每条记录代表一次评分
2. 按 user_id 分组统计 COUNT(*)
3. 按评分次数降序
4. 若次数相同，按用户名字升序
5. LIMIT 1 取第一

第二问：2 月评分最高的电影

1. 过滤 created_at 在 2020-02 月内
2. 按 movie_id 分组求 AVG(rating)
3. 按平均分降序

4. 并列按电影标题升序
5. LIMIT 1

合并输出

必须使用 UNION ALL：

- UNION 会自动去重
- 如果用户名和电影名相同，会丢掉一行

示例解法

代码块

```
(
SELECT u.name AS results
FROM MovieRating r
JOIN Users u ON r.user_id = u.user_id
GROUP BY r.user_id
ORDER BY COUNT(*) DESC, u.name ASC
LIMIT 1
)
UNION ALL
(
SELECT m.title AS results
FROM MovieRating r
JOIN Movies m ON r.movie_id = m.movie_id
WHERE r.created_at BETWEEN '2020-02-01' AND '2020-02-29'
GROUP BY r.movie_id
ORDER BY AVG(r.rating) DESC, m.title ASC
LIMIT 1
);
```

四、1321 – 餐厅增长率（滑动窗口平均）

https://leetcode.cn/problems/restaurant-growth/?envType=study-plan-v2&envId=sql-free-50

题目描述

Customer 表记录每天的消费金额 amount，要求输出：

- 从第 7 天开始
- 每一天对应过去连续 7 天的总收入和平均收入

核心模型

> 时间序列 + 滑动窗口（Rolling Window）

解题思路（详细拆解）

1. 原表可能同一天有多条记录
2. 需要先按 visited_on 汇总每日总收入
3. 对每日收入做窗口计算：
   - 最近 7 天总和
   - 最近 7 天平均值
4. 窗口范围是：ROWS 6 PRECEDING（包含当前行共 7 行）
5. 前 6 天无法构成完整窗口，需要过滤

示例解法

代码块

```
SELECT
    visited_on,
    SUM(amount) OVER (ORDER BY visited_on ROWS 6 PRECEDING) AS amount,
    ROUND(AVG(amount) OVER (ORDER BY visited_on ROWS 6 PRECEDING), 2) AS
average_amount
FROM (
    SELECT visited_on, SUM(amount) AS amount
    FROM Customer
    GROUP BY visited_on
) t
WHERE visited_on >= (
    SELECT MIN(visited_on) + INTERVAL 6 DAY FROM Customer
);
```

易错点

- 忘记先按日期聚合
- 窗口范围写错导致不是 7 天

五、602 – 好友申请 II（社交网络统计）

https://leetcode.cn/problems/friend-requests-ii-who-has-the-most-friends/?envType=study-plan-v2&envId=sql-free-50

题目描述

RequestAccepted 表记录好友关系：

- requester_id
- accepter_id

要求找出好友数最多的人。

核心模型

> **无向边统计（Undirected Graph Degree）**

解题思路（详细拆解）

1. 每条记录表示一次好友关系建立
2. requester 和 accepter 都应该各自好友数 +1
3. 因此需要把两列“展开”为同一列 id
4. 使用 UNION ALL 拼成一张单列表
5. 按 id 分组统计出现次数
6. 取出现次数最大的人

示例解法

代码块

```sql
SELECT id, COUNT(*) AS num
FROM (
    SELECT requester_id AS id FROM RequestAccepted
    UNION ALL
    SELECT accepter_id AS id FROM RequestAccepted
) t
GROUP BY id
ORDER BY num DESC
LIMIT 1;
```

易错点

- 使用 UNION 会去重，好友数会少算

六、585 – 投票最多的候选人（TopN + 排序）

https://leetcode.cn/problems/investments-in-2016/?envType=study-plan-v2&envId=sql-free-50

题目描述

Vote 表记录每一票投给的候选人 candidateId，要求输出票数最多的候选人。

解题思路（详细拆解）

1. 每条记录代表一票
2. 按 candidateId 分组
3. COUNT(*) 得到票数
4. 按票数降序排序
5. LIMIT 1 取第一名

示例解法

代码块

```sql
SELECT candidateId
FROM Vote
GROUP BY candidateId
ORDER BY COUNT(*) DESC
LIMIT 1;
```

本题模型

> 最简单的 Top1 聚合排序

七、185 – 部门工资前三高的员工（TopN per Group）

https://leetcode.cn/problems/department-top-three-salaries/?envType=study-plan-v2&envId=sql-free-50

题目描述

Employee 表包含员工工资和部门编号，要求输出每个部门工资前三高的员工。

核心难点

- Top3 不是全局，而是“每个部门内部 Top3”
- 这类题叫：TopN per Group

解题思路（详细拆解）

1. 先把员工和部门表 JOIN，拿到部门名称

2. 在每个部门内部按工资降序排名
3. 使用 DENSE_RANK：
    - 并列工资排名相同
    - 不跳号
4. 过滤排名 ⩽ 3 的记录

推荐解法

代码块

```sql
SELECT department, employee, salary
FROM (
    SELECT
        d.name AS department,
        e.name AS employee,
        e.salary,
        DENSE_RANK() OVER (
            PARTITION BY e.departmentId
            ORDER BY e.salary DESC
        ) AS rk
    FROM Employee e
    JOIN Department d
    ON e.departmentId = d.id
) t
WHERE rk <= 3;
```

易错点

- 用 ROW_NUMBER 会导致并列工资只保留一个
- 忘记 PARTITION BY 就变成全局排名

八、模型总结

题号	核心模型
1978	自连接层级关系
626	奇偶行交换
1341	UNION ALL 语义
1321	滑动窗口平均
602	无向关系统计
585	Top1 排序
185	分组 TopN

> 当你能识别题目属于哪个模型，SQL 就变成了可复用的“模板语言”，刷题效率会大幅提升。