力扣高频 SQL 50 题阶段总结（三）

本文继续按照**“不是给答案，而是讲清楚为什么这么写”的原则，总结 LeetCode 高频 SQL 50 中另一组极具代表性的题目**：

> **2356 / 1141 / 1084 / 596 / 1729 / 619 / 1045**

这一组题目覆盖了 SQL 面试中最常考的几类模型：

- 去重统计（COUNT DISTINCT）
- 连续行为判断（窗口 / 自连接）
- LEFT JOIN + 缺失数据
- 覆盖 / 买全类问题

如果说前两阶段是在练“统计能力”，那么这一阶段是在考：

> **你是否真正理解“集合”和“关系”的含义。**

一、2356. 每位教师所教授的科目种类数（Number of Unique Subjects Taught by Each Teacher）

题目描述

表 `Teacher`：

| teacher_id | subject_id | dept_id |

要求：

> 对每个 `teacher_id`，统计他教授过的 **不同科目数量**。

解题思路（详细）

这是一道**非常基础但极其容易写错的题**，因为它考察的是：

> 你是否真正理解 `COUNT(*)` 和 `COUNT(DISTINCT ...)` 的区别

① 统计维度是谁？

题目明确要求的是：

```
代码块
```

```
每个 teacher_id 一行
```

因此：

- `teacher_id` 必须出现在 `GROUP BY`

---

② 统计的对象是什么？

不是：

- 教学记录条数

而是：

- **不同的 subject_id**

如果一个老师：

- 在不同学期
- 或不同部门
- 教同一门课

👉 **只能算 1 门。**

---

③ 结论

这道题的本质就是一句话：

> **分组 + 去重计数**

解法

代码块

```sql
SELECT
  teacher_id,
  COUNT(DISTINCT subject_id) AS cnt
FROM Teacher
GROUP BY teacher_id;
```

二、1141. 查询近 30 天活跃用户数（User Activity for the Past 30 Days I）

题目描述

表 `Activity`：

| user_id | session_id | activity_date | activity_type |

要求：

> 统计 **最近 30 天内（含当天）** 每天的 **活跃用户数**。

解题思路（详细）

这道题是**时间窗口 + 去重统计**的典型题。

① 什么叫“活跃用户”？

题目定义得非常清楚：

> **在某一天有至少一条活动记录的用户**

所以：

- 一个用户一天内无论有多少条记录
- **只能算 1 次**

👉 必须用 `COUNT(DISTINCT user_id)`。

---

② 时间窗口如何处理？

- 最近 30 天
- 含 `2019-07-27`

因此：

```
代码块
1   activity_date BETWEEN '2019-06-28' AND '2019-07-27'
```

---

③ 为什么不用子查询？

因为：

- 时间过滤发生在行级
- 聚合发生在天级

一个 SELECT 就够了。

解法

代码块

```
SELECT
  activity_date AS day,
  COUNT(DISTINCT user_id) AS active_users
FROM Activity
WHERE activity_date BETWEEN '2019-06-28' AND '2019-07-27'
GROUP BY activity_date;
```

三、1084. 销售分析 III（Sales Analysis III）

题目描述

给定：

- `Product(product_id, product_name, unit_price)`
- `Sales(sale_id, product_id, sale_date, quantity)`

要求：

> 找出 **只在 2019-01-01 ~ 2019-03-31 期间销售过的商品**。

解题思路（详细，重难点）

这是“**只在某区间出现过**”的经典题型。

① 错误直觉

很多人会写：

代码块

```
WHERE sale_date BETWEEN '2019-01-01' AND '2019-03-31'
```

❌ 这是错的，因为：

- 它只保证“出现过”
- 并不能保证“**没有在区间外出现**”

---

② 正确建模方式

我们真正要判断的是：

> **某个 product 的所有 sale_date，都落在这个区间内**

翻译成 SQL 条件就是：

```
代码块
1   MIN(sale_date) >= start
2   AND MAX(sale_date) <= end
```

---

③ 为什么要 GROUP BY?

因为：

- 判断的是“产品整体行为”
- 而不是“某一条销售记录”

解法

```
代码块
1   SELECT p.product_id, p.product_name
2   FROM Product p
3   JOIN Sales s
4     ON p.product_id = s.product_id
5   GROUP BY p.product_id
6   HAVING
7     MIN(s.sale_date) >= '2019-01-01'
8     AND MAX(s.sale_date) <= '2019-03-31';
```

四、596. 超过 5 名学生的课（Classes More Than 5 Students）

题目描述

表 `Courses`：

| student | class |

要求：

> 找出 **学生人数 ≥ 5** 的课程。

解题思路（详细）

这是一道**HAVING 的入门经典题**。

① 统计维度

- 每个 `class` 一行

② 统计指标

- 不同学生数

⚠ 注意：

- 如果一个学生重复选同一门课
- 只能算一次

因此：

> **必须 COUNT(DISTINCT student)**

解法

代码块

```sql
SELECT class
FROM Courses
GROUP BY class
HAVING COUNT(DISTINCT student) >= 5;
```

五、1729. 求关注者的数量（Find Followers Count）

题目描述

表 `Followers`：

| user_id | follower_id |

要求：

> 对每个用户，统计他的关注者数量。

解题思路（详细）

这是**关系表建模的基础题**。

① 谁是统计主体？

- 被关注的人：`user_id`

② 统计什么？

- 有多少人关注他

👉 即：

```
代码块
1   COUNT(follower_id)
```

如果一个人重复关注（理论上不应该）：

- 也可以加 DISTINCT 提高健壮性

解法

```
代码块
1   SELECT
2     user_id,
3     COUNT(follower_id) AS followers_count
4   FROM Followers
5   GROUP BY user_id
6   ORDER BY user_id;
```

六、619. 只出现一次的最大数字（Biggest Single Number）

题目描述

表 `MyNumbers`：

| num |

要求：

> 找出 **只出现一次的最大数字**。

解题思路（详细）

这道题结合了三件事：

- 分组
- 过滤
- 排序 / 聚合

① 如何判断“只出现一次”？

- `GROUP BY num`
- `HAVING COUNT(*) = 1`

② 如何取“最大”？

- 对满足条件的 num
- 取 `MAX(num)`

解法

代码块

```sql
SELECT MAX(num) AS num
FROM MyNumbers
WHERE num IN (
  SELECT num
  FROM MyNumbers
  GROUP BY num
  HAVING COUNT(*) = 1
);
```

七、1045. 买下所有产品的客户（Customers Who Bought All Products）

题目描述

- `Customer(customer_id, product_key)`
- `Product(product_key)`

要求：

> 找出 **买过所有不同产品** 的客户。

解题思路（详细，重点题）

这是 SQL 面试中**极其高频的“覆盖类问题”**。

① 错误但常见的理解

> 买的数量 = 产品数量

❌ 错在：

- 数量 ≠ 种类

---

② 正确建模

我们要比较的是：

```
客户买过的【不同 product_key 数】
==
Product 表中的【不同 product_key 数】
```

---

③ 关键技术点

- `COUNT(DISTINCT product_key)`
- `HAVING` 子句

解法

```sql
SELECT customer_id
FROM Customer
GROUP BY customer_id
```

```
  HAVING COUNT(DISTINCT product_key) = (
    SELECT COUNT(DISTINCT product_key)
    FROM Product
  );
```

八、阶段总结（非常重要）

通过这 7 道题，你已经系统性地接触到了：

- 去重统计的语义边界
- GROUP BY + HAVING 的真正用途
- “只在区间内出现”的建模方式
- 覆盖 / 买全类问题的固定套路

一句话总结这一阶段：

> **SQL 的难点不在语法，而在“你在对哪个集合做比较”。**

如果你愿意，下一步我可以：

- 把你目前刷过的所有 SQL 题抽象成 **10 个面试必背模型**
- 或继续总结 **高频 SQL 50 的下一组题**