SQL语法

MySQL操作

```
-u：指定用户名
-p：指定密码
-h：主机
-P：端口

G：打印结果垂直显示
c：取消当前MySQL命令
q：退出MySQL连接
s：显示服务器状态
h：帮助信息
d：改变执行符
```

语法点

查看当前日期

```
#获取当前日期
SELECT CURRENT_DATE();
```

关联查询

```
INNER JOIN
LEFT JOIN/RIGHT JOIN
FULL JOIN（MySQL不支持，可以LEFT JOIN UNION RIGHT JOIN）
UNION：效率更高
UNION ALL：不会合并重复的记录行
```

拷贝表

```
insert into b(a, b, c) select d,e,f from a;
```

并集、差集、交集

```
Union
Except
InterSect
```

- 文章

LeetCode SQL

- 题库

595. 大的国家

- 不要使用OR，导致索引失效
- 涉及到多个列时候，每次只能选取一个索引，另一个table-scan
- UNION问题：对结果进行排序，降低性能，最好尝试后进行比较

```
select username from users where company = 'bbc' or company = 'itv';

select username from users where company IN ('bbc', 'itv');

select username from users where company = 'bbc' or city = 'London';

select username from users where company = 'bbc'
union
select username from users where city = 'London';
```

1757. 可回收且低脂的产品

584. 寻找用户推荐人

- MySQL使用三值逻辑：TRUE, FALSE 和 UNKNOWN
- NULL需要特殊判断，`IS NULL` 和 `IS NOT NULL`
- != <>

183. 从不订购的客户

- id NOT IN
- NOT EXISTS
- 区别讲解

```
SELECT
FROM
LEFT JOIN
ON                          //连接条件
WHERE xx IS NULL            //删除不匹配on后面条件的记录
```

- LEFT JOIN ON WHERE IS NULL学习

1873. 计算特殊奖金

- IF(, ,)
- regexp正则表达式
- CASE 列名 WHEN 条件 THEN 条件为真设置的值 ELSE 条件为假设置的值 END

627. 变更性别

- IF(, ,)
- CASE 列名 WHEN 条件 THEN 条件为真设置的值 ELSE 条件为假设置的值 END

196. 删除重复的电子邮箱

```
DELETE
FROM 表

DELETE
FROM 表
```

- 自己联自己，产生笛卡尔积结果，然后直接筛选临时结果看起来很好很方便 但是如果条件区分度不够，数据又不少，直接会组合出相当大的临时表 （自己测试：650条记录，6种种类做类似Email的条件区分，一下子查出1W5的临时数据， 而实际使用中很容易就是万为单位的数据量，直接会爆炸的！若说仅仅是学习了解，入门而已，那当我没说！）

1667. 修复表中的名字

```
# CONACT(str1,str2) 连接      拼接两个字符串
# UPPER(str)  大写       字符串大写
# LOWER(str)  小写       字符串小写
# LENGTH(str)  长度      字符串长度
# SUBSTRING(str,start,end)    截取   截取字符串,start开始,end结束。
# LEFT(str,len)     截取   从左边开始截取字符串
# RIGHT(str,len)     截取   从右边开始截取字符串

Oracle
initcap()：开头第一个大写，剩下小写


```

1484. 按日期分组销售产品

- GROUP_CONCAT()：在有GROUP_BY的查询语句中，将同一个分组中的值连接起来，返回一个字符串结果

1527. 患某种疾病的患者

```
LIKE：
RLIKE：搭配正则表达式（和REGEXP类似）
REGEXP
```

- 正则表达式

```
^：以xxx开头
|：或
.：不包括换行和回车的任意字符
*：重复前面的字符任意次
.*：任意个任意字符
\s：空格
```

1965. 丢失信息的雇员

```
UNION ALL 联合多表
GROUP BY 根据xxx分组
HAVING 条件筛选（GROUP BY之后可以用HAVING语句对结果进行过滤）
```

```
4 ORDER BY 排序
```

1795. 每个产品在不同商店的价格

```
1 UNION ALL 列转行用
2 CASE WHEN 行转列用
3 不考虑去重，尽量用union all
4 行列谁多了，就是谁转谁
```

608. 树节点

```
1 CASE 列名
2     WHEN 条件
3     THEN 值
4     WHEN 条件
5     THEN 值
6     END AS 列名
```

176. 第二高的薪水

- 重点是为空时候返回NULL
- 方法一：变成子查询
- 方法二：IFNULL

```
1 LIMIT 两种写法
2 LIMIT x, x
3 LIMIT 1 OFFSET 1
4
5 IF(expr1, expr2, expr3)：如果expr1为true，返回expr2；否则返回expr3
6 IFNULL(expr1,expr2)：expr1不为NULL，返回expr1；否则返回expr2
7 NULLIF(expr1,expr2)：如果expr1 = expr2，返回NULL；否则返回expr1
8 ISNULL(expr)：expr为NULL，返回1；否则返回0
```

175. 组合两个表

```
1 INNER JOIN：仅保存都有的列
2 LEFT JOIN：保留左表中的所有记录
3 RIGHT JOIN：保留右表中的所有记录
```

1581. 进店却未进行过交易的顾客

```
1 SELECT
2    customer_id, COUNT(customer_id) count_no_trans
3 FROM
4     visits v
5 LEFT JOIN
6     transactions t ON v.visit_id = t.visit_id
7 WHERE transaction_id IS NULL
```

```
GROUP BY customer_id

#连接、筛选、分组、计数
```

- 连接、筛选、分组、计数

1148. 文章浏览 I

- DISTINC对 SELECT的列进行去重

197. 上升的温度

```
INNER JOIN / JOIN 表 ON 条件
```

```
DATEDIFF(expr1,expr2)

DATEDIFF () 返回 expr1 - expr2，表示为从一个日期到另一个日期的以天为单位的值。expr1 和 expr2 是日期或日期和时间表达式。计算中仅使用值的日期部分。
```

607. 销售员

- 笛卡尔积：CROSS JOIN（m*n条记录）

```
SELECT
    s.name
FROM
    salesperson s
WHERE
    s.sales_id NOT IN (SELECT
            o.sales_id
        FROM
            orders o
                LEFT JOIN
            company c ON o.com_id = c.com_id
        WHERE
            c.name = 'RED')
```

1141. 查询近30天活跃用户数

```
WHERE DATEDIFF("2019-07-27", activity_date) BETWEEN 0 AND 29
BETWEEN包含两边边界
```

1693. 每天的领导和合伙人

```
GROUP X, Y
```

1729. 求关注者的数量

586. 订单最多的客户

511. 游戏玩法分析 I

```
GROUP BY 存留的是第一条结果
MIN取最小值
```

1890. 2020年最后一次登录

```
日期函数:
YEAR() = xxx
```

1741. 查找每个员工花费的总时间

```
SUM
GROUP BY x,y
```

1393. 股票的资本损益

```
SELECT stock_name,
    SUM(CASE operation
    WHEN 'BUY'
    THEN -price
    ELSE price
    END) AS capital_gain_loss
FROM Stocks
GROUP BY stock_name;

SELECT stock_name, SUM(IF(operation = 'BUY', -price, price)) AS capital_gain_los
FROM Stocks
GROUP BY stock_name;
```

1407. 排名靠前的旅行者

```
JOIN WHERE
LEFT JOIN ON
```

1158. 市场分析 I

182. 查找重复的电子邮箱

```
GROUP BY
HAVING count(x) > 1
```

1050. 合作过至少三次的演员和导演

1587. 银行账户概要 II

1084. 销售分析III

1532. 最近的三笔订单

- 题面

```
row_number() over(partition by xxx order by xxx)
给整张表新加一行列号，每个根据partition by的列从1开始递增
rank() over(partition by xxx order by xxx)

两者区别：
row_number()有3条排在第1位时，排序为：1，2，3，4……
rank()有3条排在第1位时，排序为：1，1，1，4……
```