公司sql面试题

1-货拉拉面试题

1.1-统计薪水最高员工

货拉拉EMPLOYEE表里包含着所有货拉拉的员工信息，每个员工有自己的工号、姓名、薪水、和部门ID

ID	Name	Salary	DepartmentID
1	Joe	70000	1
2	Henry	80000	2
3	Jane	60000	2
4	Ben	90000	1

货拉拉DEPARTMENT表里包含着货拉拉所有的部门

ID	Department
1	IT
2	Sales

现在需要统计出货拉拉每个部门薪水最高的员工，如下图

Department	Name	Salary
IT	Ben	90000
Sales	Henry	80000

数据准备：

```
#创建 EMPLOYEE 表
CREATE TABLE `EMPLOYEE`  (
  `ID` varchar(255),
  `Name` varchar(255),
  `Salary` varchar(255),
  `DepartmentID` varchar(255)
) ;
#插入数据
INSERT INTO `EMPLOYEE` VALUES ('1', 'Joe', '70000', '1');
INSERT INTO `EMPLOYEE` VALUES ('2', 'Henry', '80000', '2');
INSERT INTO `EMPLOYEE` VALUES ('3', 'Jane', '60000', '2');
INSERT INTO `EMPLOYEE` VALUES ('4', 'Ben', '90000', '1');
#创建 DEPARTMENT 表
CREATE TABLE `DEPARTMENT`  (
  `ID` varchar(255),
  `DEPARTMENT` varchar(255)
) ;
#插入数据
INSERT INTO `DEPARTMENT` VALUES ('1', 'IT');
INSERT INTO `DEPARTMENT` VALUES ('2', 'Sales');
```

```
-- 易错点：select查询的字段出现非聚合字段，非group by需要分组的字段
select department,name,max(salary),
from employee e
inner join department d
    on e.DepartmentID = d.ID
group by d.ID

# 错误点：salary没有关联部门id
select department,name,salary
from employee e
inner join department d
    on e.DepartmentID = d.ID
where salary in (select max(salary) from employee group by departmentid)

-- 现在需要统计出货拉拉每个部门薪水最高的员工
-- 方法一：
select department,name,salary
from employee e
inner join department d
    on e.DepartmentID = d.ID
inner join (select departmentid,max(salary) max_salary from employee group by
departmentid) t
    on d.ID = t.departmentid and t.max_salary = e.salary

-- 方法二：窗口（通用性更好，可以考虑一个部门里薪水最高的员工有多个）
# 专用窗口函数
-- RANK函数
-- 计算排序时，如果存在相同位次的记录，则会跳过之后的位次。
-- 例）有 3 条记录排在第 1 位时：1 位、1 位、1 位、4 位……
-- DENSE_RANK函数
-- 同样是计算排序，即使存在相同位次的记录，也不会跳过之后的位次。
-- 例）有 3 条记录排在第 1 位时：1 位、1 位、1 位、2 位……
-- ROW_NUMBER函数
-- 赋予唯一的连续位次。
-- 例）有 3 条记录排在第 1 位时：1 位、2 位、3 位、4 位……

-- rank() over() 排序
select department,name,salary,
rank() over(partition by department order by salary desc) rk
from employee e
inner join department d
    on e.DepartmentID = d.ID

-- 筛选rank=1
select
department,name,salary
from (
select department,name,salary,
rank() over(partition by department order by salary desc) rk
from employee e
inner join department d
    on e.DepartmentID = d.ID
) t
where rk=1
```

https://www.bilibili.com/video/BV1Rh411375N?p=3(修正版)

1.2-改变相邻座位

小马是货拉拉的行政同学，她有一张座位表，储存员工名字和与他们相对应的座位号id。其中第一列的id 是连续递增的因工作需要小马想改变相邻俩同事的座位。你能不能帮她写一个 SQL query 来输出小马想要的结果呢?

示例:

Seat:

ID	EMPLOYEE
1	Jack
2	Scott
3	Emery
4	Green
5	Linda
6	Jane

假如数据输入的是上表，则输出结果如下:

ID	EMPLOYEE
1	Scott
2	Jack
3	Green
4	Emery
5	Jane
6	Linda

数据准备:

```
#创建 Seat 表
CREATE TABLE `Seat`  (
  `ID` varchar(255),
  `EMPLOYEE` varchar(255)
) ;
#插入数据
INSERT INTO `Seat` VALUES ('1', 'Jack');
INSERT INTO `Seat` VALUES ('2', 'Scott');
INSERT INTO `Seat` VALUES ('3', 'Emery');
INSERT INTO `Seat` VALUES ('4', 'Green');
INSERT INTO `Seat` VALUES ('5', 'Linda');
INSERT INTO `Seat` VALUES ('6', 'Jane');
```

参考答案

```
# 只符合当前题目提供的数据集结果
select
case when mod(id,2)=0 then id-1 else id+1 end id,employee
from seat
order by id asc

# 如果最后一个id是奇数，那么不用id+1
# 分情况-- 求总id个数，最后一位数不换，其余当奇数时-1，当偶数时+1。
 select (case
       when mod(id,2)!=0 and id!=counts then id+1
       when mod(id,2)!=0 and id=counts then id
       else id-1 end)as id,EMPLOYEE
       from seat,(select count(*)as counts from seat)as seat_counts
                 order by id;
```

1.3-用户取消率

Trips表中存所有货拉拉的行程信息。每段行程有唯一键 Id，Client_Id 和 Driver_Id。Status 是枚举类型，枚举内容为 (‘completed’,‘cancelled_by_driver’,‘cancelled_by_client’)。

ID	Client_ID	Driver_ID	City_ID	Status	Request_at
1	1	10	1	completed	2019/10/1
2	2	11	1	cancelled_by_driver	2019/10/1
3	3	12	6	completed	2019/10/1
4	4	13	6	cancelled_by_driver	2019/10/1
5	1	10	1	completed	2019/10/2
6	2	11	6	completed	2019/10/2
7	3	12	6	completed	2019/10/2
8	2	12	12	completed	2019/10/3
9	3	10	12	completed	2019/10/3
10	4	13	12	cancelled_by_driver	2019/10/3

Users 表存所有用户。每个用户有唯一键 Users_Id。Banned 表示这个用户是否被禁止，Role 则是一个表示（'client', 'driver', 'partner'）的枚举类型。

Users_ID	Banned	Role
1	No	client
2	Yes	client
3	No	client
4	No	client
10	No	driver
11	No	driver
12	No	driver
13	No	driver

写一段 SQL 语句查出 2019年10月1日 至 2019年10月3日 期间非禁止用户的取消率。基于上表，你的 SQL 语句应返回如下结果，取消率（Cancellation Rate）保留两位小数。

Day	Cancellation_Rate
2019/10/1	0.33
2019/10/2	0
2019/10/3	0.5

数据准备：

```
#创建 Trips 表
CREATE TABLE `Trips`  (
  `ID` varchar(255),
  `Client_ID` varchar(255) ,
  `Driver_ID` varchar(255),
  `City_ID` varchar(255) ,
  `Status` varchar(255) ,
  `Request_at` varchar(255)
) ;
#插入数据
INSERT INTO `Trips` VALUES ('1', '1', '10', '1', 'completed', '2019-10-01');
INSERT INTO `Trips` VALUES ('2', '2', '11', '1', 'cancelled_by_driver', '2019-10-01');
INSERT INTO `Trips` VALUES ('3', '3', '12', '6', 'completed', '2019-10-01');
INSERT INTO `Trips` VALUES ('4', '4', '13', '6', 'cancelled_by_driver', '2019-10-01');
INSERT INTO `Trips` VALUES ('5', '1', '10', '1', 'completed', '2019-10-02');
INSERT INTO `Trips` VALUES ('6', '2', '11', '6', 'completed', '2019-10-02');
INSERT INTO `Trips` VALUES ('7', '3', '12', '6', 'completed', '2019-10-02');
INSERT INTO `Trips` VALUES ('8', '2', '12', '12', 'completed', '2019-10-03');
INSERT INTO `Trips` VALUES ('9', '3', '10', '12', 'completed', '2019-10-03');
INSERT INTO `Trips` VALUES ('10', '4', '13', '12', 'cancelled_by_driver', '2019-10-03');
```

```
#创建 Users 表
CREATE TABLE `Users` (
  `Users_ID` varchar(255) ,
  `Banned` varchar(255) ,
  `Role` varchar(255)
) ;
#插入数据
INSERT INTO `Users` VALUES ('1', 'No', 'client');
INSERT INTO `Users` VALUES ('2', 'Yes', 'client');
INSERT INTO `Users` VALUES ('3', 'No', 'client');
INSERT INTO `Users` VALUES ('4', 'No', 'client');
INSERT INTO `Users` VALUES ('10', 'No', 'driver');
INSERT INTO `Users` VALUES ('11', 'No', 'driver');
INSERT INTO `Users` VALUES ('12', 'No', 'driver');
INSERT INTO `Users` VALUES ('13', 'No', 'driver');
```

参考答案：

```
-- 取消率
    --2019年10月1日至2019年10月3日期间非禁止用户的的取消次数 / 非禁止用户的的所有状态(完成、未完成)
select
Request_at Day,
count(*),
sum(if(status='completed',0,1)),
round(sum(if(Status='completed',0,1))/count(*) ,2) Cancellation_Rate
from users u
left join trips t
    on u.Users_ID = t.Client_ID
where banned = 'NO' and Request_at between '2019-10-01' and '2019-10-03'
group by Request_at
```

2-北京新东方数据中心数据面试题

SQL基础测试

employee1：

职工ID	职工姓名	入职年份	部门ID
A1	B1	2000	C1
A2	B2	1998	C2
A3	B3	1999	C3
A4	B4	2001	C4

dept1表：

部门ID	部门名称	部门经理
C1	D1	E1
C2	D2	E2
C3	D3	E3
C4	D4	E4

emp1表：

职工ID	岗位ID	岗位名称
A1	G1	Gn1
A1	G2	Gn2
A2	G1	Gn1
A3	G3	Gn3
A3	G5	Gn5
A4	G4	Gn4

```
创建 Employee1 表：
CREATE TABLE `employee1`  (
  `职工ID` varchar(255),
  `职工姓名` varchar(255),
  `入职年份` varchar(255),
  `部门ID` varchar(255)
);
插入数据：
INSERT INTO `employee1` VALUES ('A1', 'B1', '2000', 'C1');
INSERT INTO `employee1` VALUES ('A2', 'B2', '1998', 'C2');
INSERT INTO `employee1` VALUES ('A3', 'B3', '1999', 'C3');
INSERT INTO `employee1` VALUES ('A4', 'B4', '2001', 'C4');

创建 Dept1 表：
CREATE TABLE `dept1`  (
  `部门ID` varchar(255),
  `部门名称` varchar(255),
  `部门经理` varchar(255)
);
插入数据：
INSERT INTO `dept1` VALUES ('C1', 'D1', 'E1');
INSERT INTO `dept1` VALUES ('C2', 'D2', 'E2');
INSERT INTO `dept1` VALUES ('C3', 'D3', 'E3');
INSERT INTO `dept1` VALUES ('C4', 'D4', 'E4');

创建 Emp1 表：
CREATE TABLE `emp1`  (
  `职工ID` varchar(255),
  `岗位ID` varchar(255),
  `岗位名称` varchar(255)
);
```

```
插入数据：
INSERT INTO `emp1` VALUES ('A1', 'G1', 'Gn1');
INSERT INTO `emp1` VALUES ('A1', 'G2', 'Gn2');
INSERT INTO `emp1` VALUES ('A2', 'G1', 'Gn1');
INSERT INTO `emp1` VALUES ('A3', 'G3', 'Gn3');
INSERT INTO `emp1` VALUES ('A3', 'G5', 'Gn5');
INSERT INTO `emp1` VALUES ('A4', 'G4', 'Gn4');
```

查询入职年份在2000年及以后的职工

```
select *
from employee1
where 入职年份 >=2000;
```

查询出部门C1的所有职工信息，需要所有的职工信息和部门信息

```
select a.*,b.部门名称,b.部门经理
from employee1 a
inner join dept1 b
    on a.部门ID=b.部门ID
where a.部门ID= 'C1';
```

查询出与A1不是一个岗位的所有职工的职工信息

```
-- 职工A1的岗位
select  岗位id  from emp1 where  职工id = 'A1';
-- 不是职工A1的岗位的职工
select  distinct  职工ID from emp1 where 岗位id not in (select  岗位id  from emp1
where  职工id = 'A1')
-- 查询出与A1不是一个岗位的所有职工的职工信息
select a.*
from
employee1 a
inner join (select  distinct 职工ID from emp1 where 岗位id not in (select  岗位id
 from emp1 where  职工id = 'A1')) b
    on a.职工id= b. 职工ID;
```

3-小红书数据分析笔试题

题目一：计算好评率

好评率是用户对产品评价的重要指标。

现在需要统计2019年3月1日到2019年3月31日，用户'小张'提交的"母婴"类目"DW"品牌的好评率（好评率=“好评”评价量/总评价量），有用户评价详情表、商品详情表：请写出SQL语言查询语句:

解题思路：

1. 好评率=“好评”评价量/总评价量
2. 要求好评率，则要使用where条件筛选指定商品的情况
3. 又因为条件要指定商品名，所以要两个表做关联

```
# 创建数据表
```

```
create table redbk_userjudge(
  id   int,
  create_time  varchar(30),
  user_name   varchar(50),
  goods_id    int,
  sub_time    varchar(30),
  sat_name   varchar(30)
);

create table redbk_goodsinfo(
  goods_id   int,
  goods_name   varchar(30),
  brand_name    varchar(30)
);

insert into redbk_userjudge values(1,'2019-01-06','你是一个大萝
卜',110307701,'2019-01-06 09:36:07','好评');
insert into redbk_userjudge values(2,'2019-03-13','小张',110400901,'2019-03-13
12:26:07','好评');
insert into redbk_userjudge values(3,'2019-03-17','武动乾坤',120308079,'2020-03-
19 15:15:07','好评');
insert into redbk_userjudge values(4,'2019-04-09','世纪大骗子',160300804,'2020-04-
19 10:28:07','中评');
insert into redbk_userjudge values(5,'2019-07-13','国际最无聊人',111507701,'2020-
07-19 23:09:07','差评');
insert into redbk_userjudge values(6,'2019-08-06','小兔子好',111607761,'2020-08-
19 17:18:07','好评');

insert into redbk_goodsinfo values(110307701,'高乐婴幼儿奶粉','DW');
insert into redbk_goodsinfo values(110400901,'母婴','DW');
insert into redbk_goodsinfo values(120308079,'潮流服饰','AD');
insert into redbk_goodsinfo values(160300804,'永坚三角木架','YJD');
insert into redbk_goodsinfo values(111507701,'恬恬咖啡豆','TP');
insert into redbk_goodsinfo values(111607761,'三只兔子坚果小零食','RB');
```

```
-- 1、计算好评率
-- 好评率是用户对产品评价的重要指标。现在需要统计2019年3月1日到2019年3月31日，用户'小张'提交
的"母婴"类目"DW"品牌的好评率（好评率=“好评”评价量/总评价量）
-- 请写出SQL语言查询语句:

select
u.user_name,u.sub_time,
sum(if(u.sat_name = '好评',1,0))/count(*) 好评率
from redbk_userjudge u
inner join redbk_goodsinfo g
    on u.goods_id = g.goods_id
where u.user_name = '小张' and g.goods_name='母婴' and g.brand_name='DW' and
u.sub_time between '2019-03-01' and '2019-03-31'
```

题目二：用户行为分析

有订单事务表、收藏事务表,要求：请用一句SQL取出所有用户对商品的行为特征，特征分为已购买、购买未收藏、收藏未购买、收藏且购买

解答思路:

1. 题目是要得到 user_id 和 item_id 的购买和收藏的组合情况，所以要使用这两个主键进行关联
2. 根据是否关联上就可以知道是否购买和是否收藏
3. full join 全外连接可以得到结果，但是mysql要使用 union 来实现

参考答案:

```
# 数据准备
create table redbk_orders(
  id   int ,
  user_id   varchar(10),
  item_id    int,
  par_time  varchar(30),
  item_num  int
);
insert into redbk_orders values(1,'001','201','2018-08-31 00:00:01',1);
insert into redbk_orders values(2,'002','203','2018-09-02 12:00:02',2);
insert into redbk_orders values(3,'003','203','2018-09-01 00:00:01',1);
insert into redbk_orders values(4,'003','203','2018-09-04 09:10:30',1);

create table redbk_favorites(
  id  int,
  user_id   varchar(10),
  item_id   int,
  fav_time   varchar(30)
);

insert into redbk_favorites values(1,'001',201,'2018-08-31 00:00:01');
insert into redbk_favorites values(2,'002',202,'2018-09-02 12:00:02');
insert into redbk_favorites values(3,'003',204,'2018-09-01 00:00:01');
```

```
-- 参考答案
select
  o.user_id,
  o.item_id,
    1 as '已购买',
  case when f.item_id is null then 1 else 0 end as '购买未收藏',
    0 as '收藏未购买',
  case when f.item_id is not null  then 1  else 0 end as '收藏且购买'
from redbk_orders o
left join redbk_favorites f
  on o.user_id=f.user_id and o.item_id=f.item_id
union
select
  f.user_id,
  f.item_id,
    case when o.item_id is not null then 1 else 0 end as  '已购买',
  0 as '购买未收藏',
    case when o.item_id is null then 1 else 0 end as '收藏未购买',
    case when o.item_id is not null then 1 else 0 end as '收藏且购买'
from redbk_favorites f
left join redbk_orders o
  on o.user_id=f.user_id and o.item_id=f.item_id
order by user_id,item_id;
```

4-滴滴数据分析笔试题

问题：

1. 提取2020年8月各城市每天的快车司机数、快车订单量和快车流水数据。
2. 提取2020年8月和9月，每个月的北京市新老司机（首单日期在当月为新司机）的司机数、在线时长和TPH（订单量/在线时长）数据。
3. 分别提取司机数大于20，司机总在线时长大于2小时，订单量大于1的城市名称数据。

现有四张表，分别是“司机数据”表，“订单数据”表，“在线时长数据”表，“城市匹配数据”表。

1、“司机数据”表，记录了日期、司机id、城市id、首次完成订单时间

日期	司机id	城市id	首次完成订单时间
2020/8/1	1	100000	2016/1/1
2020/8/1	2	100000	2018/7/2
2020/8/1	3	100000	2020/7/8

2、“订单数据”表，记录了日期、订单id、司机id、乘客id、产品线id、流水

产品线id: 1是表示专车，2表示企业，3表示快车，4表示企业快车

日期	订单id	司机id	乘客id	产品线id	流水
2020/8/1	1001	1	301	1	200
2020/8/1	1002	1	302	1	100
2020/8/1	1003	2	302	1	120

3、“在线时长数据”表，记录了日期、司机id、在线时长

日期	司机id	在线时长
2020/8/1	1	2
2020/8/1	2	1.1
2020/8/1	2	5

4、“城市匹配数据”表，记录了城市id、城市名称

城市id	城市名称
100000	北京
200000	上海
300000	天津

1、提取2020年8月各城市每天快车的司机数、快车订单量和快车流水数据

解题思路： 涉及对多表的查询，首先明确所查询的字段都来源于什么表

a.对题目关键词作判断，各自匹配哪个字段

2020年8月--日期、城市--城市id/城市名称、司机数--司机id、快车--产品线id、订单量--订单id、流水数据--流水

再辨别这些字段分别属于哪些表，寻找合适的连接字段，把相应的表联结在一起

```
-- 易错点：多张表存在相同字段时，应注意选取跟题意相匹配的字段
SELECT a.`日期`,c.`城市名称`,a.`司机id`,b.`产品线id`,b.`订单id`,b.`流水`
FROM 司机数据 a
inner JOIN 订单数据 b
    ON a.`司机id` = b.`司机id`
inner JOIN 城市匹配数据 c
    ON a.`城市id` = c.`城市id`
-- 修正
SELECT b.`日期`,c.`城市名称`,a.`司机id`,b.`产品线id`,b.`订单id`,b.`流水`
FROM 司机数据 a
inner JOIN 订单数据 b
    ON a.`司机id` = b.`司机id`
inner JOIN 城市匹配数据 c
    ON a.`城市id` = c.`城市id`
```

b.对日期，产品id作筛选、按城市作分组，对司机数、订单数、流水做聚合操作

代码如下：

```
SELECT t.`日期`,t.`城市名称`,t.产品线id,
                count(distinct t.`司机id`) AS 司机数,
                count(t.`订单id`) AS 快车订单量,
                SUM(t.`流水`) AS 快车流水数据
FROM (
        SELECT b.`日期`,c.`城市名称`,a.`司机id`,b.`产品线id`,b.`订单id`,b.`流水`
        FROM 司机数据 a
        inner JOIN 订单数据 b
            ON a.`司机id` = b.`司机id`
        inner JOIN 城市匹配数据 c
            ON a.`城市id` = c.`城市id`
        ) t
WHERE date_format(t.日期,'%Y-%m')='2020-08' AND t.`产品线id`='3'
GROUP BY t.`城市名称`,t.`日期`;
```

2、提取2020年8月和9月，每个月的北京市新老司机（首单日期在当月为新司机）的司机数、在线时长和TPH（订单量/在线时长）数据。

解题思路：

a.这题的关键在于怎么对新老司机做查询

新老司机：用if做判断,当日期=首次完成订单时间，那么为新司机，否则为老司机。

代码如下

```
# 首单日期在当月为新司机，用DATE_FORMAT(date,'%Y-%m')格式化处理
SELECT a.*,if(DATE_FORMAT(a.`日期`, '%Y-%m') = DATE_FORMAT(a.`首次完成订单时间`,
'%Y-%m'), '新司机', '老司机') AS 新老司机
FROM `司机数据` a
WHERE (a.`日期` BETWEEN '2020-08-01' AND '2020-09-30') AND a.`城市id`='100000';
```

b.最终代码

```
-- 为什么不连接表一起计算新老司机数量、订单量、在线时长
-- 原因：1、由于司机一天内可能会接多个订单，若直接关联这两张表，司机id将不唯一，会导致新老司机
那一列出现计数时出现重复，算的新老司机的数量会不准确
-- 2、在线时长表中没给出每个订单所花的时间，只给出当天所有订单共花的时间

with temp1 as ( # 求新老司机的数量
    SELECT t.`城市id`,
    MONTH(t.`日期`) AS 月份,
    SUM(IF(新老司机='新司机',1,0)) AS 新司机数,
    SUM(IF(新老司机='老司机',1,0)) AS 老司机数
    FROM
    (SELECT a.*,if(DATE_FORMAT(a.`日期`, '%Y-%m') = DATE_FORMAT(a.`首次完成订单时间
`, '%Y-%m'), '新司机', '老司机') AS 新老司机
    FROM `司机数据` a
    WHERE (a.`日期` BETWEEN '2020-08-01' AND '2020-09-30') AND a.`城市
id`='100000'
    ) t
    GROUP BY MONTH(t.`日期`)),
    temp2 as( # 求订单量
    SELECT
  MONTH(a.`日期`) AS 月份,
  count(a.`订单id`) AS 订单量
    FROM `订单数据` a
    left join `司机数据` b on a.司机id = b.司机id
    WHERE (a.`日期` BETWEEN '2020-08-01' AND '2020-09-30') AND b.`城市
id`='100000'
    GROUP BY MONTH(a.`日期`)
    ),
    temp3 as ( # 求在线时长
    SELECT
  MONTH(a.`日期`) AS 月份,
  SUM(a.`在线时长`) AS 在线时长
    FROM `在线时长数据` a
    left JOIN `司机数据` b
        ON a.`司机id`=b.`司机id`
    WHERE (a.`日期` BETWEEN '2020-08-01' AND '2020-09-30') AND b.`城市
id`='100000'
    GROUP BY MONTH(a.`日期`)
    )
select temp1.城市id,temp1.月份,temp1.新司机数,temp1.老司机数,temp2.订单量,temp3.在线时
长,temp2.订单量/temp3.在线时长 TPH
from temp1
inner join temp2
    on temp1.月份 = temp2.月份
inner join temp3
    on temp3.月份 = temp2.月份
```

3、分别提取司机数大于20，司机总在线时长大于2小时，订单量大于1的城市名称数据

解题思路：

a.司机数大于20的城市名称

司机数的计算用count(司机id)，用到的是“司机数据”表，城市名称在 “城市匹配数据”中，关联在一起即可再分组筛选即可

代码如下：

```
select a.城市id,a.城市名称,b.司机人数
from `城市匹配数据` a
inner join (
            select 城市id,count(*) 司机人数
            from `司机数据`
            group by 城市id
            having count(*)>20 ) b
    on a.城市id = b.城市id
```

b.司机总在线时长大于2小时的城市名称

总在线时长用sum(在线时长)来计算，用的是 “在线时长数据”表，而这个表中没有城市id，因此我们需要先联结“司机数据”表，得到城市id，再通过联结“城市匹配数据”表，得到对应的城市名称。

代码如下：

```
select c.*,d.总在线时长 from 城市匹配数据 c
inner join(
            select a.*,b.总在线时长
            from 司机数据  a
            inner join (
                select 司机id,sum(在线时长) 总在线时长
                from 在线时长数据
                group by 司机id
                having sum(在线时长)>2) b
                        on a.司机id = b.司机id
                ) d
    on c.城市id = d.城市id
```

c.订单量大于1的城市名称

同理b的解题思路

代码如下：

```
select c.*,d.订单数量 from 城市匹配数据 c
inner join(
            select a.*,b.订单数量
            from 司机数据 a
            inner join (
                        select 司机id,count(*) 订单数量
                        from 订单数据
                        group by 司机id
                        having count(*)>1 ) b
                    on a.司机id = b.司机id ) d
        on c.城市id = d.城市id
```

知识点总结

1.试题重点要考察的是表的联接。当题目中涉及到多个表之间的关系时，我们要找到多个表之间是通过什么条件关联的，然后进行多表关联。

2.考查如何将复杂问题拆解为简单问题的能力，把复杂的题目拆解下一个个小的问题去逐一解决

5-网易数据分析笔试题

背景： 用户分析是电商数据分析中重要的模块，在对用户特征深度理解和用户需求充分挖掘基础上，进行全生命周期的运营管理（拉新—>活跃—>留存—>价值提升—>忠诚）

一、用户复购率计算

现在数据库中有一张用户交易表 netease_order，其中有orderid（订单ID）、userid（用户ID）、amount（消费金额）、paytime（支付时间）

1.1、请写出对应的SQL语句，查出每个用户第一单的消费金额。

1.2、请写出对应的SQL语句，查出每个月的新客数（新客指在严选首次支付的用户），当月有复购的新客数，新客当月复购率（当月有复购的新客数/月总新客数）。

1.1、请写出对应的SQL语句，查出每个用户第一单的消费金额

解题思路：

1、对每一个用户的订单数据按时间进行排序

2、取出时间最小的用户订单数据

实现方法：

法一：使用窗口函数或者变量进行排序取最小值

法二：使用分组取出最小值

参考答案：

```
查出每个用户第一单的消费金额
--使用窗口函数：
select
  a.userid,
  a.amount
from
(select
   * ,
   row_number() over(partition by userid order by paytime) as rn
 from netease_order
 ) as a
where a.rn=1

--使用变量
select
  b.*
from(
select
  a.*,
```

```
    case when @user=userid then @rn:=@rn+1
         when @user:=userid then @rn:=1 end as rn
from netease_order a,(select @user:=null,@rn:=0) t
order by a.userid,paytime ) b
where b.rn=1
order by b.userid

--使用分组求最小值的方法
--1、查出第一单消费的时间
select
  t.userid,
  t.first_time,
  b.amount
from (select
  userid,
  min(paytime) as first_time
from netease_order group by userid) t
inner join netease_order b
on t.userid=b.userid
and t.first_time=b.paytime
```

1.2、请写出对应的SQL语句，查出每个月的新客数（新客指在严选首次支付的用户），当月有复购的新客数，新客当月复购率（当月有复购的新客数/月总新客数）

参考答案：

```
with temp1 as ( # 1、找出每个用户第一次下单的时间和年月
select userid,min(paytime),date_format(min(paytime),'%Y-%m') '年月'
from netease_order
group by userid
),
    temp2 as(# 2、找出每月的新客户数目
    select userid,年月,count(userid) as '新客户数'
    from
    temp1
    group by 年月
    ),
    temp3 as( # 3.1、找出当月有复购的新客户数目，第一步找出用户在第一次下单的那个月的购买次数
    select n.userid,count(n.orderid) as '次数',年月
    from netease_order n
    inner join temp1
        on n.userid = temp1.userid and temp1.年月 = date_format(paytime,"%Y-%m")
    group by temp1.userid
    ),
    temp4 as( # 3.2、找出当月有复购的新客户数目,第二步是将有复购的新客筛选出来（购买次数
>=2）
    select 年月,temp3.userid,count(temp3.userid) as '有复购的新客户数目'
    from temp3
    where 次数>=2
    group by temp3.年月
    ),
    temp5 as ( # 计算复购率
    select temp2.年月,temp2.新客户数,ifnull(temp4.有复购的新客户数目,0),
    ifnull(temp4.有复购的新客户数目/temp2.新客户数,0) 复购率
```

```
    from temp2
    left join temp4
        on temp2.年月 = temp4.年月
    ) select * from temp5
```

6-阿里支付宝数据分析sql题目

现有交易数据表 **user_sales_table**

user_name（用户名）	pay_amount（用户支付金额）

问题：求支付金额在前 20%的用户

输出要求如下：

user_name	pay_amount

```
-- ----------------------------
-- Table structure for user_sales_table
-- ----------------------------
DROP TABLE IF EXISTS `user_sales_table`;
CREATE TABLE `user_sales_table`  (
  `user_name` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NULL
DEFAULT NULL,
  `pay_amount` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci
NULL DEFAULT NULL
);
-- ----------------------------
-- Records of user_sales_table
-- ----------------------------
INSERT INTO `user_sales_table` VALUES ('动力机车', '8393');
INSERT INTO `user_sales_table` VALUES ('图灵学院诸葛', '1302');
INSERT INTO `user_sales_table` VALUES ('黑马666', '989');
INSERT INTO `user_sales_table` VALUES ('幸运Y', '129045');
INSERT INTO `user_sales_table` VALUES ('橙子', '12774');
INSERT INTO `user_sales_table` VALUES ('六点的太阳', '581985');
INSERT INTO `user_sales_table` VALUES ('小样mia', '75817');
INSERT INTO `user_sales_table` VALUES ('Eclife', '90592');
INSERT INTO `user_sales_table` VALUES ('Q仔幸运鹅', '7475');
INSERT INTO `user_sales_table` VALUES ('甜糯糯', '37194');
INSERT INTO `user_sales_table` VALUES ('樱桃丸小子', '532959');
INSERT INTO `user_sales_table` VALUES ('开水白菜', '4432');
INSERT INTO `user_sales_table` VALUES ('momo', '13734');
INSERT INTO `user_sales_table` VALUES ('粽子酱', '515');
INSERT INTO `user_sales_table` VALUES ('皮卡先生', '859501');
INSERT INTO `user_sales_table` VALUES ('社会我哥', '68868');
INSERT INTO `user_sales_table` VALUES ('凉风kk', '41647');
INSERT INTO `user_sales_table` VALUES ('脆弱羊驼', '6928');
INSERT INTO `user_sales_table` VALUES ('邻家王子', '4039');
INSERT INTO `user_sales_table` VALUES ('MK_ZZ', '315');
```

参考答案：


```
-- 思路：
-- 借助窗口函数 ntile 将每个用户和对应的支付金额分成 5 组（这样每组就有 1/5），取分组排名第一的用户组即前支付金额在前 20%的用户。
select
t.user_name,t.all_pay_amount
from (
            select
            user_name,
            sum(pay_amount) all_pay_amount,
            ntile(5) over(order by sum(pay_amount) desc) n
            from user_sales_table
            group by user_name ) t
where t.n = 1
```

7-中金财富证券互联网金融部面试题

问题要求：

写sql语句，统计：每个客户，在签约订单前（不包含签约的当日），在每个页面的访问次数，在每个点击事件的触发次数。

提示和说明：

1）所统计的是：客户在签约（购买）投顾服务订单之前所访问的各个页面的次数+在客户在签约（购买）投顾服务订单之前所点击的各个事件的次数。

2） 某人的页面访问次数的统计：在页面访问表（page_visit_test_201909）中的任意一条数据，代表某客户（“客户编号”字段）在某日（“页面访问的时间”字段）访问了某页面（“访问的页面”字段）1次，计数为1。

3） 某人点击事件次数的统计：在点击事件表（event_visit_test_201909）中的任意一条数据，代表某客户（“客户编号”字段区分）在某日（“事件发生时间”字段区分）点击了某事件（“事件名称”字段区分）1次，计数为1。

4） 最终展示将每个客户访问各页面的次数和点击各事件的次数同时展示在一个结果表中，展示形式如下：

Message Result 1 Profile Status

客户编号	找投顾	选组合	投顾观点详情页	投顾观点评论页	投顾观点列表	投顾个人页	投顾组合评论页	投顾组合详情页	互联网投顾首页	是否购买
30247447	1	1	1	0	0	1	0	0	0	1
10177017	2	1	2	0	0	0	0	0	0	1

5） 最后一个字段“是否购买”的判断条件是：如果在订单数据中能够找到对应的客户（编号），且“组合名称”不为空，则代表该客户签约（购买）了投顾服务（产品）。

6） 如果遇到重复购买的人，则第一次购买前的行为数据为：9月1日至第一次购买前的行为数据（统计）；第二次购买前的行为数据为：第一次购买和第二次购买之间用户行为数据（统计）；以此类推。

7） 上述sql完成后，需在mysql中能直接运行。

8） 提交的文档中除sql语句外，将应聘者跑出的结果截图附上。

参考答案

```
-- 统计：每个客户，在签约订单前（不包含签约的当日），在每个页面的访问次数，在每个点击事件的触发次数
select * from page_visit_test_201909 limit 10;
select * from event_visit_test_201909 limit 10;
select * from order_test_201909;

-- 有什么页面类型
select 访问的页面
from page_visit_test_201909
group by 访问的页面;

-- 有什么事件类型
select 事件名称
from event_visit_test_201909
group by 事件名称;

-- 时间维度：9月份的页面访问数据
select 页面访问时间
from page_visit_test_201909
group by 页面访问时间

-- 时间维度：9月份的事件数据
select 事件发生时间
from event_visit_test_201909
group by 事件发生时间

-- 筛选出在9月份之后真正签约的客户
select * from
order_test_201909
where 签约日期 >= '2019-09-01' and 组合名称 is not null
order by 客户编号, 签约日期;
-- 区分复购和非复购的用户
select
*,row_number() over(partition by 客户编号 order by 签约日期) rn
from(
select * from
order_test_201909
where 签约日期 >= '2019-09-01' and 组合名称 is not null
order by 客户编号, 签约日期
) t
order by 客户编号, 签约日期


-- 第一次购买前的行为数据为：9月1日至第一次购买前的行为数据（统计）
select
*
from (
                select
                *,row_number() over(partition by 客户编号 order by 签约日期) rn
                from(
                select * from
                order_test_201909
                where 签约日期 >= '2019-09-01' and 组合名称 is not null
                order by 客户编号, 签约日期
                ) t
                order by 客户编号, 签约日期
) o
```

```
inner join page_visit_test_201909 p
    on o.客户编号 = p.客户编号
where rn=1

-- 行转列
select
o.客户编号,
sum(if(访问的页面='找投顾' and 页面访问时间<签约日期, 1, 0)) as 找投顾,
sum(if(访问的页面='选组合' and 页面访问时间<签约日期, 1, 0)) as 选组合,
sum(if(访问的页面='投顾观点详情页' and 页面访问时间<签约日期, 1, 0)) as 投顾观点详情页,
sum(if(访问的页面='投顾观点评论页' and 页面访问时间<签约日期, 1, 0)) as 投顾观点评论页,
sum(if(访问的页面='投顾观点列表' and 页面访问时间<签约日期, 1, 0)) as 投顾观点列表,
sum(if(访问的页面='投顾个人页' and 页面访问时间<签约日期 ,1, 0)) as 投顾个人页,
sum(if(访问的页面='投顾组合评论页' and 页面访问时间<签约日期, 1, 0)) as 投顾组合评论页,
sum(if(访问的页面='投顾组合详情页' and 页面访问时间<签约日期, 1, 0)) as 投顾组合详情页,
sum(if(访问的页面='互联网投顾首页' and 页面访问时间<签约日期, 1, 0)) as 互联网投顾首页,
1 as 是否购买
from (
                select
                *,row_number() over(partition by 客户编号 order by 签约日期) rn
                from(
                select * from
                order_test_201909
                where 签约日期 >= '2019-09-01' and 组合名称 is not null
                order by 客户编号, 签约日期
                ) t
                order by 客户编号, 签约日期
) o
inner join page_visit_test_201909 p
    on o.客户编号 = p.客户编号
where rn=1
group by o.客户编号


-- 第二次购买前的行为数据为：第一次购买和第二次购买之间用户行为数据（统计）
# 添加字段，订单1的签约日期，订单2的签约日期
select *,
if(rn=1,签约日期,null) 第一次签约日期,
lag(if(rn=1,签约日期,null),1) over(order by 客户编号, 签约日期) 第一次签约日期_,
if(rn=2,签约日期,null) 第二次签约日期
from(
        select
        *,row_number() over(partition by 客户编号 order by 签约日期) rn
        from(
        select * from
        order_test_201909
        where 签约日期 >= '2019-09-01' and 组合名称 is not null
        order by 客户编号, 签约日期
        ) t
        order by 客户编号, 签约日期
) t1

-- 第一次购买和第二次购买之间用户的行为特征数据
select
o.客户编号,
sum(if(访问的页面='找投顾' and 页面访问时间<签约日期, 1, 0)) as 找投顾,
sum(if(访问的页面='选组合' and 页面访问时间<签约日期, 1, 0)) as 选组合,
sum(if(访问的页面='投顾观点详情页' and 页面访问时间<签约日期, 1, 0)) as 投顾观点详情页,
```

```
sum(if(访问的页面='投顾观点评论页' and 页面访问时间<签约日期, 1, 0)) as 投顾观点评论页,
sum(if(访问的页面='投顾观点列表' and 页面访问时间<签约日期, 1, 0)) as 投顾观点列表,
sum(if(访问的页面='投顾个人页' and 页面访问时间<签约日期, 1, 0)) as 投顾个人页,
sum(if(访问的页面='投顾组合评论页' and 页面访问时间<签约日期, 1, 0)) as 投顾组合评论页,
sum(if(访问的页面='投顾组合详情页' and 页面访问时间<签约日期, 1, 0)) as 投顾组合详情页,
sum(if(访问的页面='互联网投顾首页' and 页面访问时间<签约日期, 1, 0)) as 互联网投顾首页,
1 as 是否购买
from (
            select *,
            if(rn=1,签约日期,null) 第一次签约日期,
            lag(if(rn=1,签约日期,null),1) over(order by 客户编号, 签约日期) 第一次签约
日期_,
            if(rn=2,签约日期,null) 第二次签约日期
            from(
                    select
                    *,row_number() over(partition by 客户编号 order by 签约日期) rn
                    from(
                    select * from
                    order_test_201909
                    where 签约日期 >= '2019-09-01' and 组合名称 is not null
                    order by 客户编号, 签约日期
                    ) t
                    order by 客户编号, 签约日期
            ) t1
) o
inner join page_visit_test_201909 p
    on o.客户编号 = p.客户编号
where rn=2 and 页面访问时间 between 第一次签约日期_ and 第二次签约日期
group by o.客户编号
```

行业数据分析sql面试题

1-游戏类面试题1

1.1

请写出一个查询，得到同时满足下列三个条件的游戏类型

（1）游戏上线时间在2013年到2016年之间

（2）游戏类型(game_type)为storybook和food making

（3）game_type 的游戏数量大于2

Table name: **app_info**

app_name	live_date	game_type
Episode - Choose Your Story	2013/12/18	storybook
Kim Kardashian: Hollywood	2014/6/24	storybook
Super Estilista	2018/11/22	beauty salon
Monster High™ Beauty Shop	2017/10/16	beauty salon
Choices: Stories You Play	2016/6/22	storybook
Slime Simulator Relax Games	2019/10/19	food making
JoJo Siwa - Live to Dance	2019/8/29	combination games
Shave Salon Spa Games	2018/7/5	beauty salon
DIY Fashion Star	2018/6/13	beauty salon
Girls Hair Salon	2014/9/25	beauty salon

数据准备：

```
#创建 app_info 表
CREATE TABLE `app_info`  (
  `app_name` varchar(255) ,
  `live_date` varchar(255) ,
  `game_type` varchar(255)
) ;
#插入数据
INSERT INTO `app_info` VALUES ('Episode - Choose Your Story', '2013-12-18',
'storybook');
INSERT INTO `app_info` VALUES ('Kim Kardashian: Hollywood', '2014-06-24',
'storybook');
INSERT INTO `app_info` VALUES ('Super Estilista', '2018-11-22', 'beauty salon');
INSERT INTO `app_info` VALUES ('Monster High™ Beauty Shop', '2017-10-16',
'beauty salon');
INSERT INTO `app_info` VALUES ('Choices: Stories You Play', '2016-06-22',
'storybook');
INSERT INTO `app_info` VALUES ('Slime Simulator Relax Games', '2019-10-19',
'food making');
INSERT INTO `app_info` VALUES ('JoJo Siwa - Live to Dance', '2019-08-29',
'combination games');
INSERT INTO `app_info` VALUES ('Shave Salon Spa Games', '2018-07-05', 'beauty
salon');
INSERT INTO `app_info` VALUES ('DIY Fashion Star', '2018-06-13', 'beauty
salon');
INSERT INTO `app_info` VALUES ('Girls Hair Salon', '2014-09-25', 'beauty
salon');
```

参考答案：

```
（1）游戏上线时间在2013年到2016年之间
（2）游戏类型(game_type)为storybook和food making
（3）game_type 的游戏数量大于2。


SELECT game_type
FROM app_info
WHERE YEAR(live_date)  BETWEEN 2013 AND 2016
AND game_type IN ('storybook','food making')
GROUP BY game_type
HAVING COUNT(1) > 2
```

1.2

题意：将数据生成 年、月、及对应每月对应实际数据（题目中的数据为累加，例如需统计2016年2月数据，需要将1月份给扣减）

Question 2:

table: promotion_data　　Only accumulated impression is available, not daily data

log_date	accumulated_impressions
2015-12-31	900
2016-01-01	1,000
2016-01-02	1,100
2016-01-03	1,200
2016-01-04	1,300
2016-01-05	1,400
...	...
2016-01-29	3,800
2016-01-30	3,900
2016-01-31	4,000
2016-02-01	4,100
2016-02-02	4,200
2016-02-03	4,300

Question: Write a SQL to calcualte monthly impressions. For the above tables, your SQL query should return the following rows

请注意：这里的数据都是累计数，不是每天的数.比如2015-12-31，accumulated_impressions=900 表示从最开始（时间未知）到2015-12-31这一天累计的impression数为900.

请写出SQL以后自行计算一下结果是否与下表相符

Year	Month	Monthly Impressions
2016	1	3,100
2016	2	300

数据准备：

```
#创建 promotion_data 表
CREATE TABLE `promotion_data`(
  `log_date` varchar(255),
  `accumulated_impressions` varchar(255)
);
#插入数据
INSERT INTO `promotion_data` VALUES ('2015-12-31', '900');
INSERT INTO `promotion_data` VALUES ('2016-01-01', '1000');
INSERT INTO `promotion_data` VALUES ('2016-01-02', '1100');
INSERT INTO `promotion_data` VALUES ('2016-01-03', '1200');
INSERT INTO `promotion_data` VALUES ('2016-01-04', '1300');
INSERT INTO `promotion_data` VALUES ('2016-01-05', '1400');
INSERT INTO `promotion_data` VALUES ('2016-01-06', '1500');
INSERT INTO `promotion_data` VALUES ('2016-01-07', '1600');
INSERT INTO `promotion_data` VALUES ('2016-01-08', '1700');
```

```
INSERT INTO `promotion_data` VALUES ('2016-01-09', '1800');
INSERT INTO `promotion_data` VALUES ('2016-01-10', '1900');
INSERT INTO `promotion_data` VALUES ('2016-01-11', '2000');
INSERT INTO `promotion_data` VALUES ('2016-01-12', '2100');
INSERT INTO `promotion_data` VALUES ('2016-01-13', '2200');
INSERT INTO `promotion_data` VALUES ('2016-01-14', '2300');
INSERT INTO `promotion_data` VALUES ('2016-01-15', '2400');
INSERT INTO `promotion_data` VALUES ('2016-01-16', '2500');
INSERT INTO `promotion_data` VALUES ('2016-01-17', '2600');
INSERT INTO `promotion_data` VALUES ('2016-01-18', '2700');
INSERT INTO `promotion_data` VALUES ('2016-01-19', '2800');
INSERT INTO `promotion_data` VALUES ('2016-01-20', '2900');
INSERT INTO `promotion_data` VALUES ('2016-01-21', '3000');
INSERT INTO `promotion_data` VALUES ('2016-01-22', '3100');
INSERT INTO `promotion_data` VALUES ('2016-01-23', '3200');
INSERT INTO `promotion_data` VALUES ('2016-01-24', '3300');
INSERT INTO `promotion_data` VALUES ('2016-01-25', '3400');
INSERT INTO `promotion_data` VALUES ('2016-01-26', '3500');
INSERT INTO `promotion_data` VALUES ('2016-01-27', '3600');
INSERT INTO `promotion_data` VALUES ('2016-01-28', '3700');
INSERT INTO `promotion_data` VALUES ('2016-01-29', '3800');
INSERT INTO `promotion_data` VALUES ('2016-01-30', '3900');
INSERT INTO `promotion_data` VALUES ('2016-01-31', '4000');
INSERT INTO `promotion_data` VALUES ('2016-02-01', '4100');
INSERT INTO `promotion_data` VALUES ('2016-02-02', '4200');
INSERT INTO `promotion_data` VALUES ('2016-02-03', '4300');
```

参考答案:

```
-- 将数据生成 年、月、及对应每月对应实际数据（题目中的数据为累加，例如需统计2016年2月数据，需要将1月份给扣减）


-- 1、新增一个字段data2,data2的数据是根据data1，向下移动一位得到
select
log_date,
accumulated_impressions as data1,
lag(accumulated_impressions,1) over(order by log_date asc) data2
from promotion_data

# data2-data1 获得每天实际新增
select
*,
date_format(log_date,'%Y-%m') ym,
(data1-data2) as day_increment
from (
    select
    log_date,
    accumulated_impressions as data1,
    lag(accumulated_impressions,1) over(order by log_date asc) data2
    from promotion_data
)  t1

# 分组年月，聚合每天实际新增得到每月实际新增
select
ym,
sum(day_increment) month_increment
```

```
from(
select
*,
date_format(log_date,'%Y-%m') ym,
(data1-data2) as day_increment
from (
    select
    log_date,
    accumulated_impressions as data1,
    lag(accumulated_impressions,1) over(order by log_date asc) data2
    from promotion_data
)  t1
) t2
group by ym;
```

1.3

请查询出以下表。”<7 days” 表示此玩家每一行日期的前7天之内玩过游戏(不是今天之前的前7天，是玩家玩游戏日期的前7天)

Table name: **play_time**

player_name	play_date	play_length
Charlie Campbell	2020/1/31	180
Charlie Campbell	2020/2/1	150
Marco Bates	2020/2/1	120
Nathan Bennett	2020/2/2	133
Calvin Carr	2020/2/3	160
Frederick Gill	2020/2/4	154
Marco Bates	2020/2/6	173
Frederick Gill	2020/2/12	220
Nathan Bennett	2020/2/12	210
Calvin Carr	2020/2/16	201
Charlie Campbell	2020/2/23	99

```
# 创建 play_time 表
CREATE TABLE `play_time`(
  `player_name` varchar(255),
  `play_date` date,
    `play_length` int
);

# 插入数据
INSERT INTO `play_time` VALUES ('Charlie Campbell', '2020/1/31',180);
INSERT INTO `play_time` VALUES ('Charlie Campbell', '2020/2/1',150);
```

```
INSERT INTO `play_time` VALUES ('Marco Bates', '2020/2/1',120);
INSERT INTO `play_time` VALUES ('Nathan Bennett', '2020/2/2',133);
INSERT INTO `play_time` VALUES ('Calvin Carr', '2020/2/3',160);
INSERT INTO `play_time` VALUES ('Frederick Gill', '2020/2/4',154);
INSERT INTO `play_time` VALUES ('Marco Bates', '2020/2/6',173);
INSERT INTO `play_time` VALUES ('Frederick Gill', '2020/2/12',220);
INSERT INTO `play_time` VALUES ('Nathan Bennett', '2020/2/12',220);
INSERT INTO `play_time` VALUES ('Calvin Carr', '2020/2/16',201);
INSERT INTO `play_time` VALUES ('Frederick Gill', '2020/2/23',99);
```

显示样式:

player_name	play_date	play_length	<7 days
Charlie Campbell	2020/1/31	180	no
Charlie Campbell	2020/2/1	150	yes
Marco Bates	2020/2/1	120	no
Nathan Bennett	2020/2/2	133	no
Calvin Carr	2020/2/3	160	no
Frederick Gill	2020/2/4	154	no
Marco Bates	2020/2/6	173	yes
Frederick Gill	2020/2/12	220	no
Nathan Bennett	2020/2/12	210	no
Calvin Carr	2020/2/16	201	no
Charlie Campbell	2020/2/23	99	no

```
-- 参考答案

# 添加辅助列 play_date_
select
*,
lag(play_date,1) over(partition by player_name order by play_date) play_date_
from play_time

# 判断play_date 和 play_date_ 的日期差是否小于7天
select
player_name,
play_date,
play_length,
case when play_date_ is null then 'no'
         when datediff(play_date,play_date_)<7 then 'yes'
         else 'no' end '<7 days'
from (
        select
        *,
        lag(play_date,1) over(partition by player_name order by play_date)
play_date_
        from play_time
```

```
) t
order by player_name,play_date ;
```


2-游戏类面试题2

2-不同游戏充值金额

当前有两个数据表：【game】、【income】

请计算2020年Q1季度各部门、各游戏的总充值金额，输出字段：【部门】、【游戏名】，充值金额【sum_income_money】

数据准备：

```
#创建 game 表
CREATE TABLE `game`  (
  `department` varchar(255),
  `game_name` varchar(255),
  `game_id` varchar(255)
) ;
#插入数据
INSERT INTO `game` VALUES ('业务1', '开心消消乐', '1000');
INSERT INTO `game` VALUES ('业务2', '阴阳师', '1001');
INSERT INTO `game` VALUES ('业务1', '刺激战场', '1005');
INSERT INTO `game` VALUES ('业务4', '王者荣耀', '1007');
#创建 income 表
CREATE TABLE `income`  (
  `uid` varchar(255),
  `game_id` varchar(255),
  `income_money` varchar(255),
  `income_time` varchar(255)
);
#插入数据
INSERT INTO `income` VALUES ('2333', '1000', '168', '2020-04-01');
INSERT INTO `income` VALUES ('2333', '1000', '268', '2020-04-02');
INSERT INTO `income` VALUES ('2333', '1001', '30', '2020-03-03');
INSERT INTO `income` VALUES ('2334', '1005', '6', '2020-03-04');
INSERT INTO `income` VALUES ('2336', '1005', '6', '2020-03-05');
INSERT INTO `income` VALUES ('2339', '1007', '1', '2020-03-06');
INSERT INTO `income` VALUES ('2338', '1007', '648', '2020-03-07');
INSERT INTO `income` VALUES ('2338', '1007', '648', '2020-03-08');
```

参考答案：

```
-- 请计算2020年Q1季度各部门、各游戏的总充值金额，输出字段：【部门】、【游戏名】，充值金额【sum_income_money】

-- 获取相关字段及筛选2020年第一季度数据
select a.*,department,
game_name
from income a
left join game b
    on a.game_id = b.game_id
where year(income_time)=2020 and month(income_time)<=3
```

```
-- 获取各部门、各游戏的总充值金额
select
department,
game_name,
sum(income_money) as sum_income_money
from (
        select a.*,department,
        game_name
        from income a
        left join game b
            on a.game_id = b.game_id
        where year(income_time)=2020 and month(income_time)<=3
    ) c
group by department,game_name;
```

3-电商类面试题1

3.1

商品活动流水表，表名为event，字段：goods_id， time；

求参加活动次数最多的商品的最近一次参加活动的时间

数据准备：

```
#创建 event 表
CREATE TABLE `event`  (
  `goods_id` varchar(255),
  `time` varchar(255)
  );
#插入数据
INSERT INTO `event` VALUES ('可口可乐', '2021-02-22');
INSERT INTO `event` VALUES ('雪碧', '2021-03-01');
INSERT INTO `event` VALUES ('可口可乐', '2021-03-04');
INSERT INTO `event` VALUES ('东鹏特饮', '2021-03-06');
INSERT INTO `event` VALUES ('雪碧', '2021-03-07');
INSERT INTO `event` VALUES ('红牛', '2021-03-07');
INSERT INTO `event` VALUES ('东鹏特饮', '2021-03-08');
INSERT INTO `event` VALUES ('雪碧', '2021-03-08');
```

参考答案：

```
-- 求参加活动次数最多的商品的最近一次参加活动的时间
-- 统计每种商品参加活动的次数，以及最近一次参与活动的时间
SELECT goods_id,
        max(time) as max_time,
        count(goods_id) as goods_num
FROM event group by goods_id

-- 排序次数，取最大值
SELECT * FROM (
SELECT goods_id,
        max(time) as max_time,
        count(goods_id) as goods_num
        FROM event group by goods_id) a
```

```
ORDER BY goods_num DESC
LIMIT 1;
```


3.2

现有交易数据表user_goods_table如下：

user_name	用户名
date	日期
order_number	订单号
goods_kind	用户订购的的外卖品类
num	数量

现在想知道每个用户购买最多的外卖品类是哪个。

```
#创建user_goods_table表
CREATE TABLE `user_goods_table`  (
  `user_name` varchar(255),
  `date` varchar(255),
  `order_number` varchar(255),
  `goods_kind` varchar(255),
  `num` varchar(255)
  );
#插入数据
INSERT INTO `user_goods_table` VALUES ('张三', '2021-02-03', '147450', '苹果',
'5');
INSERT INTO `user_goods_table` VALUES ('李四', '2021-02-04', '434334', '汉堡',
'2');
INSERT INTO `user_goods_table` VALUES ('王五', '2021-02-05', '433434', '香蕉',
'8');
INSERT INTO `user_goods_table` VALUES ('张三', '2021-02-04', '224432', '梨子',
'7');
INSERT INTO `user_goods_table` VALUES ('李四', '2021-03-04', '343433', '冰淇淋',
'3');
INSERT INTO `user_goods_table` VALUES ('张三', '2021-03-02', '232434', '芒果',
'4');
INSERT INTO `user_goods_table` VALUES ('王五', '2021-03-02', '342434', '面包',
'5');
```

输出要求如下：

user_name 用户名

goods_kind 该用户购买的最多外卖品类

参考答案：

```
-- 每个用户购买最多的外卖品类是哪个

-- 统计每个用户购买的外卖种类的数量
SELECT user_name,goods_kind,sum(num) sum_num
FROM user_goods_table
```

```
GROUP BY user_name, goods_kind

-- 排序不同用户购买的外卖种类的次数
SELECT user_name,goods_kind,sum_num,
row_number()over(partition by user_name order by sum_num desc) rn
FROM (
        SELECT user_name,goods_kind,sum(num) sum_num
     FROM user_goods_table
        GROUP BY user_name, goods_kind) t

-- 筛选rn=1,即是每个用户购买的最多的外卖种类
SELECT user_name, goods_kind
FROM
    (SELECT user_name,goods_kind,sum_num,
        row_number()over(partition by user_name order by sum_num desc) rn
        FROM (
        SELECT user_name,goods_kind,sum(num) sum_num
     FROM user_goods_table
        GROUP BY user_name, goods_kind) t) s
 where rn=1;
```

4-电商类面试题2

根据下面的例表编写SQL查询语句

数据准备

```
#建表
CREATE TABLE Product (
        Product varchar(10),
        Category varchar(10),
        Color varchar(10),
        Weight decimal(3, 1),
        Price int
);

CREATE TABLE Order_table (
        OrderID int,
        Name varchar(10),
        OrderDate date,
        Store varchar(10),
        Product varchar(10),
        Quantity int,
        Amount int
);

CREATE TABLE Store (
        Store varchar(10),
        City varchar(10)
);

#插入数据
INSERT INTO Product
VALUES ('ProductA', 'CategoryA', 'Yellow', 5.6,100),
        ('ProductB', 'CategoryA', 'Red', 3.7,200),
```

```
        ('ProductC', 'CategoryB', 'Blue', 10,3,300),
        ('ProductD', 'CategoryB', 'Black', 7.8,400);

INSERT INTO Order_table
VALUES (1,'CustomerA', '2018-01-01', 'StoreA','ProductA',1,100),
        (1,'CustomerA', '2018-01-01', 'StoreA','ProductB',1,200),
        (1,'CustomerA', '2018-01-01', 'StoreA','ProductC',1,300),
        (2,'CustomerB', '2018-01-12', 'StoreB','ProductB',1,200),
        (2,'CustomerB', '2018-01-12', 'StoreB','ProductD',1,400),
        (3,'CustomerC', '2018-01-12', 'StoreC','ProductB',1,200),
        (3,'CustomerC', '2018-01-12', 'StoreC','ProductC',1,300),
        (3,'CustomerC', '2018-01-12', 'StoreC','ProductD',1,400),
        (4,'CustomerA', '2018-01-01', 'StoreD','ProductD',1,800),
        (5,'CustomerB', '2018-01-23', 'StoreB','ProductA',1,100);

INSERT INTO Store
VALUES ('StoreA', 'CityA'),
        ('StoreB', 'CityA'),
        ('StoreC', 'CityB'),
        ('StoreD', 'CityC'),
        ('StoreE', 'CityD'),
        ('StoreF', 'CityB');
```

4.1

请查找符合下列要求的产品，并按照产品价格降序排列: Category为 (CategoryA 且颜色为 yellow)，或者(Weight大于5)

```
-- 请查找符合下列要求的产品，并按照产品价格降序排列: Category为 (CategoryA 且颜色为
yellow)，或者(Weight大于5)
SELECT *
FROM Product
WHERE (Category = 'CategoryA' and Color = 'Yellow') or (Weight > 5)
ORDER BY Price DESC;
```

4.2

请计算每一位客人的购买金额(Amount)，总购买订单数，总购买产品件数(Quantity)，同一个客人同一天的订单算做一单，并筛选总购买金额大于等于800的客人，按金额降序排列

```
-- 请计算每一位客人的购买金额( Amount)，总购买订单数，总购买产品件数( Quantity)，同一个客人
同一天的订单算做一单，并筛选总购买金额大于等于800的客人，按金额降序排列
SELECT * FROM (
    SELECT
        Name,
        sum(Amount) as '总购买金额',
    count(distinct OrderDate) as '总购买订单数',
    sum(Quantity) as '总购买产品件数'
    FROM Order_table
        GROUP BY Name
        ) a
where 总购买金额>800
order by 总购买金额 desc;
```

4.3

请找出每个城市(City)购买金额排名第二的客人，列出购买城市，姓名，和购买金额

```
-- 请找出每个城市(City)购买金额排名第二的客人，列出购买城市，姓名，和购买金额
SELECT
city,
Name,
sum(Amount) as 'sum_amount',
rank() over (partition by city order by sum(amount) desc) as rk
FROM (
    SELECT city,name,amount
        FROM Order_table a
    left join Store b
        on a.Store=b.Store
        ) t1
group by city,Name
order by city,sum(Amount) desc
;

SELECT *
FROM (
            SELECT
            city,
            Name,
            sum(Amount) as 'sum_amount',
            rank() over (partition by city order by sum(amount) desc) as rk
            FROM (
                    SELECT city,name,amount
                    FROM Order_table a
                    left join Store b
                            on a.Store=b.Store
                            ) t1
            group by city,Name
            order by city,sum(Amount) desc
) t2
where rk = 2;
```

视频讲解：https://www.bilibili.com/video/BV1Rh411B7sN?p=23

4.4

购买过 ProductA且购买过 ProductB的顾客人数

```
-- 购买过 ProductA且购买过 ProductB的顾客人数
SELECT count(a.Name) as '顾客人数'
FROM
     (SELECT distinct Name FROM Order_table where Product = 'ProductA') a
inner join
     (SELECT distinct Name FROM Order_table where Product = 'ProductB') b
    on a.Name = b.Name;
```

4.5

求每个分类产品有哪些颜色，要求输出颜色合并显示在一个字段内

```
-- 求每个分类产品有哪些颜色，要求输出颜色合并显示在一个字段内
SELECT
*
FROM product

-- group_concat( [distinct] 要连接的字段 [order by 排序字段 asc/desc ] [separator '分隔符'] )
SELECT
Category,
GROUP_CONCAT(distinct Color)
FROM PRODUCT
GROUP BY Category;
```

5-电商类面试题3

数据准备

```
#建表
CREATE TABLE Trans (
COUNTER_ID char(20) comment"柜台ID",
Counter_Name char(20) comment"柜台名称",
Customer_ID char(20) comment"客户ID",
Order_ID char(20) comment"订单ID",
Product_ID char(20) comment"产品ID",
Product_Name_Cn char(20) comment"产品名称",
Quantity int comment"购买数量",
Unit_Price decimal(8, 1) comment"产品单价",
purchase_date date comment"购买时间",
Brand char(20) comment"购买品牌"
);

CREATE TABLE `Profile` (
Customer_ID Char(20) comment"客户ID",
FIRST_PUR_DATE date comment"购买时间",
age int comment"客户年龄"
);

# 插入数据
INSERT INTO Trans
VALUES ('offline001', 'counterA', 'customer001','20180214001' ,'product001','SK-II神仙水',1,1590.0,'2018-02-14','SK-II'),
('offline002', 'counterB', 'customer002','20190315002' ,'product002','雅诗兰黛小棕瓶精华液',3,899.0,'2019-03-15','雅诗兰黛'),
('offline003', 'counterC', 'customer003','20190416003' ,'product003','兰蔻小黑瓶肌底精华液',2,1080.0,'2019-04-16','LANCOME'),
('offline004', 'counterD', 'customer004','20190517004' ,'product004','海蓝之谜修护精萃液',3,1150.0,'2019-05-17','LAMER'),
('offline005', 'counterE', 'customer005','20190618005' ,'product005','玉兰油淡斑小白瓶精华',10,498.0,'2019-06-18','OLAY'),
('offline001', 'counterA', 'customer001','20180501012' ,'product001','SK-II神仙水',2,1590.0,'2018-05-01','SK-II'),
('offline004', 'counterD', 'customer001','20180618065' ,'product004','海蓝之谜修护精萃液',2,1150.0,'2018-06-18','LAMER'),
```

```
('offline004', 'counterD', 'customer001','20181110065' ,'product004','海蓝之谜修护精萃液',6,1150.0,'2018-11-10','LAMER'),
('offline004', 'counterD', 'customer001','20190213033' ,'product004','海蓝之谜修护精萃液',8,1150.0,'2018-06-18','LAMER'),
('offline002', 'counterB', 'customer002','20190429088' ,'product002','雅诗兰黛小棕瓶精华液',5,899.0,'2019-04-29','雅诗兰黛'),
('online001', 'NA', 'customer002','20190616125' ,'product002','雅诗兰黛小棕瓶精华液',6,899.0,'2019-06-16','雅诗兰黛'),
('offline003', 'counterC', 'customer003','20190617015' ,'product003','兰蔻小黑瓶肌底精华液',1,1080.0,'2019-06-17','LANCOME'),
('offline004', 'counterD', 'customer004','20190901024' ,'product004','海蓝之谜修护精萃液',3,1150.0,'2019-09-01','LAMER'),
('offline005', 'counterE', 'customer005','20190818035' ,'product005','玉兰油淡斑小白瓶精华',3,498.0,'2019-08-18','OLAY'),
('offline005', 'counterE', 'customer005','20191018077' ,'product005','玉兰油淡斑小白瓶精华',5,498.0,'2019-10-18','OLAY'),
('offline002', 'counterB', 'customer005','20191110085' ,'product005','雅诗兰黛小棕瓶精华液',2,899.0,'2019-11-10','雅诗兰黛');

INSERT INTO Profile
VALUES ('customer001','2018-02-14', 22),
('customer002','2019-03-15', 25),
('customer003','2019-04-16', 28),
('customer004','2019-05-17', 33),
('customer005','2019-06-18', 26);
```

5.1

请计算每个客户在2019年的总花费(Sales)，购买的订单数(Frequency)，每单平均花费(AUS)及每单平均购买的数量(PT)

参考答案：

```
-- 请计算每个客户在2019年的总花费(Sales)，购买的订单数(Frequency)，每单平均花费(AUS)及每单平均购买的数量(PT)
SELECT Customer_ID,
sum(Quantity*Unit_Price) as'Sales',
count(Order_ID) as'Frequency',
sum(Quantity*Unit_Price)/count(Order_ID) as'AUS',
sum(Quantity)/count(Order_ID) as'PT'
FROM Trans where year(purchase_date)=2019 GROUPBY Customer_ID;
```

5.2

请计算每一类客户在2019年每月产生的销售(即购买金额大于0，及发生购买的人数(客户类型分为:新客，老客；新客:2019年产生第一次购买；老客:2019年之前已经购买过)

参考答案：

1、增加新老客标签

```
-- 请计算每一类客户在2019年每月产生的销售(即购买金额大于0，及发生购买的人数(客户类型分为:新客，老客；新客:2019年产生第一次购买；老客:2019年之前已经购买过)
--  1、增加新老客标签
select a.*,b.*,
case when year(b.FIRST_PUR_DATE)=2019 then 'new' else 'old' end as
'Customer_Type'
from Trans a
left join Profile b
    on a.Customer_ID = b.Customer_ID
where year(a.purchase_date)=2019
```

2、分类统计即可

```
-- 2、分类客户类型，统计销售额
SELECT
                DATE_FORMAT( purchase_date, '%Y-%m' ) AS 'date_ym',
        Customer_Type,
        sum( Quantity * Unit_price ) AS 'sale_money',
        count(*) AS 'shop_people'
FROM
        (
            select a.*,
            case when year(b.FIRST_PUR_DATE)=2019 then 'new' else 'old' end as
'Customer_Type'
            from Trans a
            left join Profile b
                on a.Customer_ID = b.Customer_ID
            where year(a.purchase_date)=2019
        ) t
GROUP BY MONTH ( purchase_date ),Customer_Type;
```

5.3

现要给2019年新客发送A、B小样，线上新客发放A小样，线下新客发放B小样，请编写程序抽取相应的名单

字段 COUNTERID：offline 线下，online 线上

```
-- 现要给2019年新客发送A、B小样，线上新客发放A小样，线下新客发放B小样，请编写程序抽取相应的名单
SELECT
distinct t.customer_id,
CASE WHEN t.counter_id LIKE 'online%' THEN 'A' ELSE 'B'  END '小样类型'
FROM
       (
            select a.*,
            case when year(b.FIRST_PUR_DATE)=2019 then 'new' else 'old' end as
'Customer_Type'
            from Trans a
            left join Profile b
                on a.Customer_ID = b.Customer_ID
            where year(a.purchase_date)=2019
        ) t
where t.Customer_Type = 'new'
```

5.4

展示每种品牌的销售数量，要求排序，并输出每家商铺对应占该品牌的销售额占比情况

```
-- 每种品牌的销售数量、销售额
SELECT Brand,sum(Quantity) as 'sum_quantity',sum(Quantity*Unit_Price) as'Sales'
FROM Trans GROUP BY Brand

-- 每家店铺的销售数量、销售额
SELECT Brand,Counter_Name,sum(Quantity) as
'sum_quantity',sum(Quantity*Unit_Price) as'Sales' FROM Trans GROUP BY
Brand,Counter_Name

-- 展示每种品牌的销售数量，要求排序，并输出每家商铺对应占该品牌的销售额占比情况
SELECT
a.Brand,
a.Counter_Name,
a.sales/b.sales as '销售额占比',
b.sum_quantity as '销售数量'
FROM (SELECT Brand,Counter_Name,sum(Quantity) as
'sum_quantity',sum(Quantity*Unit_Price) as'Sales' FROM Trans GROUP BY
Brand,Counter_Name) a
LEFT JOIN (SELECT Brand,sum(Quantity) as 'sum_quantity',sum(Quantity*Unit_Price)
as'Sales' FROM Trans GROUP BY Brand) b
    on a.Brand = b.Brand
ORDER BY 销售数量 desc
```

6-线上教育类面试题1

题目1

学生表: Students

Column Name	Type
student_id	int
student_name	varchar

```
create table Students (student_id int , student_name varchar(20));
insert into Students (student_id, student_name) values ('1', 'Alice');
insert into Students (student_id, student_name) values ('2', 'Bob');
insert into Students (student_id, student_name) values ('13', 'John');
insert into Students (student_id, student_name) values ('6', 'Alex');
```

科目表: Subjects

Column Name	Type
subject_name	varchar

```
Create table Subjects (subject_name varchar(20));
insert into Subjects (subject_name) values ('Math');
insert into Subjects (subject_name) values ('Physics');
insert into Subjects (subject_name) values ('Programming');
```

考试表: Examinations

Column Name	Type
student_id	int
subject_name	varchar
Column Name	Type

```
Create table Examinations (student_id int, subject_name varchar(20));
insert into Examinations (student_id, subject_name) values ('1', 'Math');
insert into Examinations (student_id, subject_name) values ('1', 'Physics');
insert into Examinations (student_id, subject_name) values ('1',
'Programming');
insert into Examinations (student_id, subject_name) values ('2',
'Programming');
insert into Examinations (student_id, subject_name) values ('1', 'Physics');
insert into Examinations (student_id, subject_name) values ('1', 'Math');
insert into Examinations (student_id, subject_name) values ('13', 'Math');
insert into Examinations (student_id, subject_name) values ('13',
'Programming');
insert into Examinations (student_id, subject_name) values ('13', 'Physics');
insert into Examinations (student_id, subject_name) values ('2', 'Math');
insert into Examinations (student_id, subject_name) values ('1', 'Math');
```

考试表的每一行记录就表示学生表里的某个学生参加了一次科目表里某门科目的测试。

6-要求写一段 SQL 语句，查询出每个学生参加每一门科目测试的次数，结果按 student_id 和 subject_name 排序。

结果表需包含所有学生和所有科目（即便测试次数为0）

Result table:

student_id	student_name	subject_name	attended_exams
1	Alice	Math	3
1	Alice	Physics	2
1	Alice	Programming	1
2	Bob	Math	1
2	Bob	Physics	0
2	Bob	Programming	1
6	Alex	Math	0
6	Alex	Physics	0

student_id	student_name	subject_name	attended_exams
6	Alex	Programming	0
13	John	Math	1
13	John	Physics	1
13	John	Programming	1

参考答案：

```
-- 要求写一段 SQL 语句，查询出每个学生参加每一门科目测试的次数，结果按 student_id 和
subject_name 排序。
select * from students;
select * from subjects;
select * from examinations;

-- coss join 关联每科课程
select *
from students st
cross join subjects su

-- 左关联，注意选取字段来源的表
select *
from students st
cross join subjects su
left join examinations e
    on st.student_id = e.student_id and su.subject_name = e.subject_name

-- 分组统计
-- count（字段）：判断字段是否为NULL，在过滤统计。不包含NULL值行
select st.student_id,su.subject_name,count(e.subject_name)
from students st
cross join subjects su
left join examinations e
    on st.student_id = e.student_id and su.subject_name = e.subject_name
group by st.student_id,su.subject_name
```

7-线上教育类面试题2

```
# 创建 TABLE1 表：
CREATE TABLE `table1`  (
  `STU_NBR` varchar(255),
  `STU_NAM` varchar(255),
  `STU_SEX` varchar(255),
  `STU_AGE` varchar(255),
  `STU_HIT` varchar(255),
  `STU_WET` varchar(255),
  `DAT_DTE` varchar(255)
);
# 插入数据：
INSERT INTO `table1` VALUES ('A01', '张三', '男', '20', '185', '70', '2019-02-
01');
INSERT INTO `table1` VALUES ('A02', '李四', '女', '21', '166', '50', '2019-01-
01');
```

```
INSERT INTO `table1` VALUES ('A03', '王五', '男', '20', '180', '69', '2018-12-
01');
INSERT INTO `table1` VALUES ('A04', '赵六', '女', '21', '172', '60', '2019-01-
01');
INSERT INTO `table1` VALUES ('A04', '赵六', '女', '20', '170', '85', '2018-10-
01');

# 创建 TABLE2 表:
CREATE TABLE `table2`  (
  `COU_NBR` varchar(255),
  `COU_NAM` varchar(255),
  `COU_TIM` varchar(255),
  `COU_SCO` varchar(255)
);
# 插入数据:
INSERT INTO `table2` VALUES ('C01', '语文', '25', '150');
INSERT INTO `table2` VALUES ('C02', '数学', '30', '150');
INSERT INTO `table2` VALUES ('C03', '英语', '28', '120');

# 创建 TABLE3 表:
CREATE TABLE `table3`  (
  `COU_NBR` varchar(255),
  `COU_NAM` varchar(255),
  `STU_NBR` varchar(255),
  `STU_NAM` varchar(255),
  `STU_SCO` varchar(255)
);
# 插入数据:
INSERT INTO `table3` VALUES ('C01', '语文', 'A01', '张三', '129');
INSERT INTO `table3` VALUES ('C01', '语文', 'A02', '李四', '132');
INSERT INTO `table3` VALUES ('C02', '数学', 'A02', '李四', '145');
INSERT INTO `table3` VALUES ('C02', '数学', 'A04', '赵六', '102');
INSERT INTO `table3` VALUES ('C03', '英语', 'A04', '赵六', '108');
```

7.1-查询 TABLE1中的全部数据

```
-- 1、查询 TABLE1中的全部数据
Select * from table1
```

7.2-统计男生、女生人数

```
-- 2、统计男生、女生人数
Select stu_sex,count(stu_sex) from table1 group by stu_sex;
```

7.3-查询同学当前的年龄

创建表 TABLE4，包含学生当前的学号、姓名、性别、年龄、身高、体重，并从

TABLE4中查询姓名为王五的同学当前的年龄

```
-- 3、创建表 TABLE4，包含学生当前的学号、姓名、性别、年龄、身高、体重，并从TABLE4中查询姓名为王五的同学当前的年龄
 drop table table4;
 Create table table4 as(
 select stu_nbr,stu_nam,stu_sex,stu_age,stu_hit,stu_wet from table1
 );
 Select stu_age from table4 where Stu_nam='王五'
```

7.4-列出总分前20%同学名单

列出总分数在前20%高的同学名单，包含学号、姓名、分数，最后一行显示全班的平均成绩

```
 -- 列出总分数在前20%高的同学名单，包含学号、姓名、分数，最后一行显示全班的平均成绩
 -- 每位同学的总分
select
t3.stu_nbr,
t3.stu_nam,
sum(t3.stu_sco) sum_stu_sco
from table3 t3
group by t3.stu_nbr

-- ntile 分5堆 取第一堆（1/5）


select
stu_nbr,
stu_nam,
sum_stu_sco
from(
                select
                t3.stu_nbr,
                t3.stu_nam,
                sum(t3.stu_sco) sum_stu_sco,
                ntile(5) over(order by sum(t3.stu_sco) desc) n
                from table3 t3
                group by t3.stu_nbr
) t
where n=1
union all
select
null,null,
sum(stu_sco)/count(distinct t3.stu_nbr) 平均分
from table3 t3
```

7.5-计算得分率

统计不同学生的语文、数学、英语分数及总得分率(所有科目得分/所有科目总分值)

```
-- 5、统计不同学生的语文、数学、英语分数及总得分率(不同同学所有科目总得分/不同同学考试所有科目总分值)
select * from table3;
with temp1 as( -- 不同学生的[语文、数学、英语分数]-行转列
            select
```

```
            STU_NBR,
            STU_NAM,
            max(case when t3.cou_nam='语文' then stu_sco end) '语文' ,
            max(case when t3.cou_nam='数学' then stu_sco end) '数学' ,
            max(case when t3.cou_nam='英语' then stu_sco end) '英语'
            from table3 t3
            group by stu_nbr
),
        temp2 as(   -- 不同学生的总得分率
        select
        STU_NBR,
        round(sum(stu_sco)/sum(cou_sco),2) 总得分率
        from table3 t3
        left join table2 t2
            on t3.COU_NBR = t2.COU_NBR
        group by t3.STU_NBR
)
select STU_NAM,语文,数学,英语,总得分率 from temp1 inner join temp2 on temp1.STU_NBR
= temp2.STU_NBR
```

8-线上教育类面试题3

```
#创建 buy_data 表
CREATE TABLE `buy_data` (
  `user_id` varchar(255),
  `buy_datetime` varchar(255),
  `course_type` varchar(255),
  `cost_num` varchar(255)
);
#插入数据
INSERT INTO `buy_data` VALUES ('1589871', '2021-01-20', '2', '256');
INSERT INTO `buy_data` VALUES ('1589871', '2021-02-20', '2', '256');
INSERT INTO `buy_data` VALUES ('1589873', '2021-01-21', '2', '256');
INSERT INTO `buy_data` VALUES ('1589874', '2021-01-22', '2', '256');
INSERT INTO `buy_data` VALUES ('1589875', '2021-01-23', '2', '256');
INSERT INTO `buy_data` VALUES ('1589876', '2021-01-24', '2', '256');
INSERT INTO `buy_data` VALUES ('1589877', '2021-01-25', '2', '256');
INSERT INTO `buy_data` VALUES ('1589878', '2021-01-26', '2', '256');
INSERT INTO `buy_data` VALUES ('1589879', '2021-01-27', '2', '256');
INSERT INTO `buy_data` VALUES ('1589880', '2021-01-20', '2', '256');
INSERT INTO `buy_data` VALUES ('1589880', '2021-02-20', '2', '256');
INSERT INTO `buy_data` VALUES ('1589880', '2021-03-20', '2', '256');
INSERT INTO `buy_data` VALUES ('1589880', '2021-04-20', '2', '256');
INSERT INTO `buy_data` VALUES ('1589880', '2021-05-20', '2', '256');
INSERT INTO `buy_data` VALUES ('1589880', '2021-06-20', '2', '256');
INSERT INTO `buy_data` VALUES ('1589880', '2021-07-20', '2', '256');
INSERT INTO `buy_data` VALUES ('1589880', '2021-08-20', '2', '256');
INSERT INTO `buy_data` VALUES ('1589880', '2021-09-20', '2', '256');
INSERT INTO `buy_data` VALUES ('1589880', '2021-10-20', '2', '256');
INSERT INTO `buy_data` VALUES ('1589881', '2021-01-25', '2', '256');
```

问题：统计不同月份用户的回购数
回购：上月购买用户中有多少用户本月又再次购买

日期	当月首购数	次月回购	三月回购	四月回购	五月回购	六月回购
2021-01	10	2	1	1	1	1
2021-02	2	1	1	1	1	1
2021-03	1	1	1	1	1	1
2021-04	1	1	1	1	1	1
2021-05	1	1	1	1	1	1
2021-06	1	1	1	1	1	0
2021-07	1	1	1	1	0	0
2021-08	1	1	1	0	0	0
2021-09	1	1	0	0	0	0
2021-10	1	0	0	0	0	0

参考答案

```
-- 问题：统计不同月份用户的回购数
-- 回购：上月购买用户中有多少用户本月又再次购买

-- left join 关联表,能关联上的就是第二个月有回购
select *
from buy_data a
left join buy_data b
    on a.user_id = b.user_id and date_format(date_sub(b.buy_datetime,interval 1
month),'%Y-%m')=date_format(a.buy_datetime,'%Y-%m')

-- 当月首购数，次月回购数(on连接)
select
date_format(a.buy_datetime,'%Y-%m') ym,
count(distinct a.user_id) 当月首购数,
count(distinct b.user_id) 次月回购数
from buy_data a
left join buy_data b
    on a.user_id = b.user_id and date_format(date_sub(b.buy_datetime,interval 1
month),'%Y-%m')=date_format(a.buy_datetime,'%Y-%m')
group by date_format(a.buy_datetime,'%Y-%m')

-- -- 当月首购数，次月回购数(case when 判断)
select
date_format(a.buy_datetime,'%Y-%m') as '日期',
count(distinct a.user_id) as '首购数',
count(distinct case when date_format(date_sub(b.buy_datetime,interval 1
month),'%Y-%m')=date_format(a.buy_datetime,'%Y-%m') then b.user_id else null
end) as 次月回购
from buy_data a
left join buy_data b
    on a.user_id = b.user_id
group by date_format(a.buy_datetime,'%Y-%m')

-- 统计不同月份的回购数
select
 date_format(a.buy_datetime,'%Y-%m') as '日期',
 count(distinct a.user_id) as '当月首购数',
 count(distinct case when date_format(date_sub(b.buy_datetime,interval 1
month),'%Y-%m')=date_format(a.buy_datetime,'%Y-%m') then b.user_id else null
end) as 次月回购,
 count(distinct case when date_format(date_sub(b.buy_datetime,interval 2
month),'%Y-%m')=date_format(a.buy_datetime,'%Y-%m') then b.user_id else null
end) as 三月回购,
```

```
count(distinct case when date_format(date_sub(b.buy_datetime,interval 3
month),'%Y-%m')=date_format(a.buy_datetime,'%Y-%m') then b.user_id else null
end) as 四月回购,
count(distinct case when date_format(date_sub(b.buy_datetime,interval 4
month),'%Y-%m')=date_format(a.buy_datetime,'%Y-%m') then b.user_id else null
end) as 五月回购,
 count(distinct case when date_format(date_sub(b.buy_datetime,interval 5
month),'%Y-%m')=date_format(a.buy_datetime,'%Y-%m') then b.user_id else null
end) as 六月回购
from buy_data a
left join buy_data b
    on a.user_id = b.user_id
group by date_format(a.buy_datetime,'%Y-%m')
 ;
```

9-地产类面试题

房源表： house_info

房源id	house_id
维护人id	hold_ucid
委托类型	del_type
维护门店编码	hold_shop_code

门店表：shop_info

门店id	shop_id
区域	area

电话信息表：telephone_info

房源id	house_id
电话拨打日期	create_date
拨通状态	call_status

9.1、求出每个区域中有人维护且委托类型为二手和新房的房源量

```
-- 创建数据表
-- 房源表
CREATE TABLE house_info (
    house_id int,
    hold_ucid int,
    del_type varchar(30),
    hold_shop_code int
);
-- 插入数据
INSERT INTO house_info (`house_id`, `hold_ucid`, `del_type`, `hold_shop_code`)
```

```
VALUES ('24', '9904', '二手', '1004'),
    ('25', '9905', '二手', '1005'),
    ('26', '9906', '新房', '1006'),
    ('27', '9907', '新房', '1007'),
    ('28', '9908', '新房', '1008');

-- 创建数据表
-- 门店信息表
CREATE TABLE shop_info (
    shop_id int,
    area varchar(30)
);
-- 插入数据
INSERT INTO shop_info (shop_id, area)
VALUES (1001, '宝安'),
    (1004, '南山'),
    (1005, '南山'),
    (1006, '宝安'),
    (1007, '南山'),
    (1008, '龙岗');

-- 准备数据：telephone_info
CREATE TABLE telephone_info (
    house_id int,
    create_date datetime,
    call_status varchar(20)
);


# 接通状态1为成功，0为不成功
INSERT INTO telephone_info (house_id, create_date, call_status)
VALUES (23, '2020-06-17', 1),
    (24, '2020-05-07', 1),
    (25, '2020-04-05', 1),
    (26, '2020-05-06', 0),
    (27, '2020-06-18', 1),
    (28, '2020-03-29', 1),
    (29, '2020-05-25', 1),
    (30, '2020-02-18', 1),
    (27, '2020-05-16', 1),
    (26, '2020-06-17', 1);
```

答案参考：

```
select * from house_info;
select * from shop_info;

-- 求出每个区域中有人维护且委托类型为二手和新房的房源量
select
area,
sum(if(h.del_type="二手",1,0)) as ershou_nums,
sum(if(h.del_type="新房",1,0)) as newhouse_nums
from shop_info s
inner join house_info h
   on s.shop_id = h.hold_shop_code
group by s.area;
```

9.2、求出近30天内未成功接通（接通状态为“0”）客户电话的所有房源（提示:最近一次接通成功记录在30天以外，或者从未有过接通成功记录的房源），及该房源最新一次成功接通电话的日期

```
-- 求出近30天内未成功接通（接通状态为“0”）客户电话的所有房源（提示:最近一次接通成功记录在30天以外，或者从未有过接通成功记录的房源），
-- 及该房源最新一次成功接通电话的日期

select * from house_info;
select * from shop_info;
select * from telephone_info;


-- 最新一次成功接通电话的日期
SELECT house_id, MAX(create_date) AS last_call
FROM telephone_info
WHERE call_status = 1
GROUP BY house_id

-- house_info left join b。如果b表的last_call有null记录，则为从未有过接通成功记录的房源
SELECT a.house_id, last_call
FROM house_info a
LEFT JOIN (
          SELECT house_id, MAX(create_date) AS last_call
          FROM telephone_info
          WHERE call_status = 1
          GROUP BY house_id
     ) b
 ON a.house_id = b.house_id

 -- 继续筛选最近一次接通成功记录在30天以外 或 从未有过接通成功记录的房源
SELECT a.house_id, last_call
FROM house_info a
LEFT JOIN (
          SELECT house_id, MAX(create_date) AS last_call
          FROM telephone_info
          WHERE call_status = 1
          GROUP BY house_id
     ) b
 ON a.house_id = b.house_id
 WHERE datediff(now(), last_call) > 30 OR last_call IS NULL;
```

10-交通行业类面试题

限时20分钟

现有 **for_vkt_calc** 和 **st_area_linkgroup** 两张表

for_vkt_calc

字段说明：

linkid：路段编号（类似于身份证号，是路段的唯一id）

volume：路段车流量

st_area_linkgroup

字段说明：

groupid：区域所属组（如行政区县、街道区域等）

areaid：区域编号（类似于身份证号，是区域的唯一id）

linkid：路段编号（类似于身份证号，是路段的唯一id）

linklength：路段长度

linkclass：路段所属的道路等级。（如：2代表主干路，5代表次支路）

问题：请利用以上两张表中的信息，编写SQL语句，计算各区域中各个道路等级的VKT比例------VMT

计算某一区域各个道路等级的VKT比例的方法如下：

第一步：计算该区域中各路段的VKT值

第二步：计算该区域中各个道路等级的VKT值

即将该区域中主干路的所有路段的VKT求和

第三步：计算该区域中各个道路等级的VKT比例-VMT

例如主干路（linkclass=2）的VKT比例为：

VMT主=VKT主 / VKT总　即该区域中主干路的VKT值除以该区域中所有道路等级的总的VKT值

数据准备：

```
#创建 for_vkt_calc 表
CREATE TABLE `for_vkt_calc`(
  `linkid` varchar(255),
  `volume` varchar(255)
  );
#插入数据
INSERT INTO `for_vkt_calc` VALUES ('101', '132.3');
INSERT INTO `for_vkt_calc` VALUES ('102', '192.1');
INSERT INTO `for_vkt_calc` VALUES ('103', '313.2');
INSERT INTO `for_vkt_calc` VALUES ('104', '123.5');
INSERT INTO `for_vkt_calc` VALUES ('105', '0');
INSERT INTO `for_vkt_calc` VALUES ('106', '312.1');
INSERT INTO `for_vkt_calc` VALUES ('107', '211');
INSERT INTO `for_vkt_calc` VALUES ('108', '235');
INSERT INTO `for_vkt_calc` VALUES ('109', '257');
INSERT INTO `for_vkt_calc` VALUES ('110', '265');
INSERT INTO `for_vkt_calc` VALUES ('111', '235.3');
INSERT INTO `for_vkt_calc` VALUES ('112', '231.2');
INSERT INTO `for_vkt_calc` VALUES ('113', '110.3');
INSERT INTO `for_vkt_calc` VALUES ('114', '93.2');
INSERT INTO `for_vkt_calc` VALUES ('115', '133');
INSERT INTO `for_vkt_calc` VALUES ('116', '51.7');
INSERT INTO `for_vkt_calc` VALUES ('117', '67.8');
INSERT INTO `for_vkt_calc` VALUES ('118', '133.2');
INSERT INTO `for_vkt_calc` VALUES ('119', '186.1');
INSERT INTO `for_vkt_calc` VALUES ('120', '231.1');
INSERT INTO `for_vkt_calc` VALUES ('121', '234');
INSERT INTO `for_vkt_calc` VALUES ('122', '16');
INSERT INTO `for_vkt_calc` VALUES ('123', '177.2');
INSERT INTO `for_vkt_calc` VALUES ('124', '131.2');
```

```
INSERT INTO `for_vkt_calc` VALUES ('125', '166.7');
INSERT INTO `for_vkt_calc` VALUES ('126', '123.1');
#创建 st_area_linkgroup 表
 CREATE TABLE `st_area_linkgroup`(
   `groupid` varchar(255),
   `linkid` varchar(255),
   `areaid` varchar(255),
   `linklength` varchar(255),
   `linkclass` varchar(255)
   );
#插入数据
INSERT INTO `st_area_linkgroup` VALUES ('1', '101', '210', '41.41', '2');
INSERT INTO `st_area_linkgroup` VALUES ('1', '102', '210', '41.41', '2');
INSERT INTO `st_area_linkgroup` VALUES ('1', '103', '210', '41.41', '5');
INSERT INTO `st_area_linkgroup` VALUES ('1', '104', '210', '155.31', '5');
INSERT INTO `st_area_linkgroup` VALUES ('1', '105', '211', '155.31', '2');
INSERT INTO `st_area_linkgroup` VALUES ('1', '106', '212', '155.31', '5');
INSERT INTO `st_area_linkgroup` VALUES ('1', '107', '211', '93.134', '2');
INSERT INTO `st_area_linkgroup` VALUES ('1', '108', '212', '515.2', '5');
INSERT INTO `st_area_linkgroup` VALUES ('1', '109', '213', '231.23', '5');
INSERT INTO `st_area_linkgroup` VALUES ('1', '110', '211', '19', '2');
INSERT INTO `st_area_linkgroup` VALUES ('1', '111', '211', '219.3', '2');
INSERT INTO `st_area_linkgroup` VALUES ('1', '112', '212', '219.3', '2');
INSERT INTO `st_area_linkgroup` VALUES ('1', '113', '213', '93.134', '5');
INSERT INTO `st_area_linkgroup` VALUES ('1', '114', '213', '93.134', '5');
INSERT INTO `st_area_linkgroup` VALUES ('1', '115', '213', '413.3', '2');
INSERT INTO `st_area_linkgroup` VALUES ('1', '116', '212', '413.3', '2');
INSERT INTO `st_area_linkgroup` VALUES ('1', '117', '212', '98.1', '5');
INSERT INTO `st_area_linkgroup` VALUES ('1', '118', '212', '98.1', '5');
INSERT INTO `st_area_linkgroup` VALUES ('1', '119', '215', '515.2', '5');
INSERT INTO `st_area_linkgroup` VALUES ('1', '120', '215', '31.3', '2');
INSERT INTO `st_area_linkgroup` VALUES ('1', '121', '213', '41.6', '2');
INSERT INTO `st_area_linkgroup` VALUES ('1', '122', '212', '31.3', '2');
INSERT INTO `st_area_linkgroup` VALUES ('1', '123', '213', '50.3', '5');
INSERT INTO `st_area_linkgroup` VALUES ('1', '124', '211', '50.3', '2');
INSERT INTO `st_area_linkgroup` VALUES ('1', '125', '215', '23.1', '5');
INSERT INTO `st_area_linkgroup` VALUES ('1', '126', '214', '21.3', '2');
```

参考答案

```
-- 现有 for_vkt_calc 和 st_area_linkgroup 两张表
-- 问题：请利用以上两张表中的信息，编写SQL语句，计算各区域中各个道路等级的VKT比例-------VMT
-- 计算某一区域各个道路等级的VKT比例的方法如下：
-- 第一步：计算该区域中各路段的VKT值
-- 第二步：计算该区域中各个道路等级的VKT值
-- 即将该区域中主干路的所有路段的VKT求和
-- 第三步：计算该区域中各个道路等级的VKT比例VMT
-- 例如主干路（linkclass=2）的VKT比例为：
-- VMT主=VKT主 / VKT总      即该区域中主干路的VKT值除以该区域中所有道路等级的总的VKT值



select * from for_vkt_calc;
select * from st_area_linkgroup;


-- VKT(vehicle kilometers of travel ):车辆的行驶公里数
```

```
-- VKT = VOLUME*LINKLENGTH
 SELECT s.*,F.volume,(s.linklength * f.volume) as VKT
 FROM st_area_linkgroup s
 left join for_vkt_calc f
    on s.linkid = f.linkid;

-- 各区域中各个道路等级的VKT
 SELECT areaid,linkclass,sum((s.linklength * f.volume)) as VKT
 FROM st_area_linkgroup s
 left join for_vkt_calc f
    on s.linkid = f.linkid
group by areaid,linkclass

-- 各区域中的VKT
 SELECT areaid,sum((s.linklength * f.volume)) as area_VKT
 FROM st_area_linkgroup s
 left join for_vkt_calc f
    on s.linkid = f.linkid
group by areaid

-- 计算各区域中各个道路等级的VKT比例-------VMT
select
a.areaid,
linkclass,
round(vkt/area_vkt,2) vmt
from(
             SELECT areaid,linkclass,sum((s.linklength * f.volume)) as VKT
             FROM st_area_linkgroup s
             left join for_vkt_calc f
                on s.linkid = f.linkid
            group by areaid,linkclass
            ) a
inner   join (SELECT areaid,sum((s.linklength * f.volume)) as area_VKT
         FROM st_area_linkgroup s
         left join for_vkt_calc f
            on s.linkid = f.linkid
        group by areaid) b
    on a.areaid = b.areaid
```

11-金融保险类面试题

数据准备:

```
#创建 data 表
CREATE TABLE `data` (
  `保单号` varchar(255),
  `结案日` varchar(255),
  `时效1` varchar(255),
  `时效2` varchar(255),
  `时效3` varchar(255)
  );
#插入数据
INSERT INTO `data` VALUES ('A935166285', '2018-01-05', '1', '5', '15');
INSERT INTO `data` VALUES ('C112498321', '2018-05-31', '2', '2', '21');
```

```
INSERT INTO `data` VALUES ('C155268441', '2018-03-15', '1', '3', '16');
INSERT INTO `data` VALUES ('A110195877', '2019-01-06', '10', '4', '11');
INSERT INTO `data` VALUES ('H110885461', '2019-02-28', '3', '2', '18');
INSERT INTO `data` VALUES ('C885198300', '2018-12-25', '1', '3', '10');
```


5.1-求不同年月的时效的平均值

```
-- 求不同年月的时效的平均值
SELECT date_format(结案日,'%Y-%m') as '月',
avg(时效1) as '时效1平均',
avg(时效2) as '时效2平均',
avg(时效3) as '时效3平均'
FROM data
GROUP BY date_format(结案日,'%Y-%m');
```

5.2-时间维度分析

若已知表中的结案日范围为2018年1月至2019年2月，请以“2018年”，“2019年1月”和“2019年2月”三个维度分别统计时效的平均值

```
SELECT date_format(结案日,'%Y年') as '时间范围',
avg(时效1) as '时效1平均',
avg(时效2) as '时效2平均',
avg(时效3) as '时效3平均'
FROM data where date_format(结案日,'%Y年')= '2018年'
union all
SELECT date_format(结案日,'%Y年%m月') as '时间范围',
avg(时效1) as '时效1平均',
avg(时效2) as '时效2平均',
avg(时效3) as '时效3平均'
FROM data
where date_format(结案日,'%Y年%m月')= '2019年01月'
union all
SELECT date_format(结案日,'%Y年%m月') as '时间范围',
avg(时效1) as '时效1平均',
avg(时效2) as '时效2平均',
avg(时效3) as '时效3平均'
FROM data where date_format(结案日,'%Y年%m月')= '2019年02月';
```

持续更新

sql面试常考知识点

1-留存率

【相机】是深受大家喜爱的应用之一，现在我们需要研究相机的活跃情况，需统计如下数据：

某日活跃的用户(uid)在后续的一周内的留存情况（计算次留、三留、七留）

指标定义：

某日活跃用户数：某日活跃的去重用户数

N日留存用户数：某日活跃的用户在之后的第N日活跃用户数

N日活跃留存率：N日留存用户数/某日活跃用户数

例：

20180501日去重用户数10000，这批用户20190503仍有7000人活跃，则3日活跃的留存率位7000/10000=70%

数据表名：act_user_info

字段名	字段类型	备注
uid	string	用户唯一标识
app_name	string	应用名称，如相机、微信等
duration	bigint	启动时长、按天汇总
times	bigint	启动次数，按天汇总
dayno	string	表分区字段，如2018=05-01

所获得的结果如下表

日期	活跃用户数	次日留存数	三日留存数	七日留存数	次日留存率	三日留存率	七日留存率
20180501	4	1	2	0	50.00%	25.00%	0.00%

要求：

1、活跃用户数为整数

2、该留存率表示为百分比，结果保留为两位小数

3、仅一条sql或hive sql完成

4、仅写出实现查询的代码即可

```
--  某日活跃的用户在后续的一周内的留存情况(计算次留、三留、七留)

-- 指标定义
-- 某日活跃用户数：某日活跃的去重用户数
--  N日留存用户数：某日活跃的用户在之后的第N日活跃用户数
--  N日活跃留存率：N日留存用户数/某日活跃用户数

-- a.计算某日活跃用户数：某日活跃的去重用户数
select dayno, count(distinct uid) as 活跃用户数
from act_user_info
where app_name='相机'
group by dayno ;


-- b.次日留存用户数:：某日活跃的用户在之后的第二日继续活跃用户数

-- 先join两子表，为了比较当日和第N日的用户信息
select
    *
from
```

```
    (select uid,date_format(dayno,'%Y%m%d')as day1 from act_user_info where
app_name='相机') a
inner join
    (select uid,date_format(dayno,'%Y%m%d')as day2 from act_user_info where
app_name='相机') b
on a.uid=b.uid

-- 作逻辑判断，筛选出第一天活跃，第二天也活跃的用户
select
    a.day1,case when day2-day1=1 then a.uid end as uid
from
    (select uid,date_format(dayno,'%Y%m%d')as day1 from act_user_info where
app_name='相机') a
inner join
    (select uid,date_format(dayno,'%Y%m%d')as day2 from act_user_info where
app_name='相机') b
on a.uid=b.uid
;

-- 根据日期分组，统计去重用户id,计算次日留存用户数
select
    a.day1,count(distinct case when day2-day1=1 then a.uid end) 次日留存用户数
from
    (select uid,date_format(dayno,'%Y%m%d')as day1 from act_user_info where
app_name='相机') a
inner join
    (select uid,date_format(dayno,'%Y%m%d')as day2 from act_user_info where
app_name='相机') b
on a.uid=b.uid
group by a.day1;

-- 三日和七日的算法是同样的逻辑

-- C.N日活跃留存率：N日留存用户数/某日活跃用户数

-- 可用concat拼接数值和百分比符号
select
    day1 日期,count(distinct a.uid) 活跃用户数,
    count(distinct case when day2-day1=1 then a.uid end) 次日留存用户数,
    count(distinct case when day2-day1=2 then a.uid end) 三日留存用户数,
    count(distinct case when day2-day1=6 then a.uid end) 七日留存用户数,
    concat(round(count(distinct case when day2-day1=1 then a.uid
end)/count(distinct a.uid),2)*100,'%') 次日留存率,
    concat(round(count(distinct case when day2-day1=2 then a.uid
end)/count(distinct a.uid),2)*100,'%') 三日留存率,
    concat(round(count(distinct case when day2-day1=6 then a.uid
end)/count(distinct a.uid),2)*100,'%') 七日留存率
from (select uid,date_format(dayno,'%Y%m%d') day1 from act_user_info where
app_name = '相机') a
inner join (select uid,date_format(dayno,'%Y%m%d') day2 from act_user_info where
app_name = '相机') b
on a.uid=b.uid
group by day1;
```

2-计算中位数、平均数、众数

请写SQL求出中位数、平均数、众数（求中位数是请考虑奇数和偶数两种情况）


```
create table emp_salary (emp_name VARCHAR(20),emp_salary varchar(20),from_date
date,to_date date);
INSERT INTO `emp_salary`(`emp_name`, `emp_salary`, `from_date`, `to_date`)
VALUES (1001, 15000, '2020-03-01', '2020-03-31');
INSERT INTO `emp_salary`(`emp_name`, `emp_salary`, `from_date`, `to_date`)
VALUES (1002, 12000, '2020-03-01', '2020-03-31');
INSERT INTO `emp_salary`(`emp_name`, `emp_salary`, `from_date`, `to_date`)
VALUES (1003, 14000, '2020-03-01', '2020-03-31');
INSERT INTO `emp_salary`(`emp_name`, `emp_salary`, `from_date`, `to_date`)
VALUES (1004, 15000, '2020-03-01', '2020-03-31');
```

2.1、求员工工资的众数

```
-- 求员工工资的众数
SELECT emp_salary,count(1) AS cnt
FROM emp_salary
GROUP BY emp_salary
HAVING count(*) >= ALL(SELECT COUNT(*) FROM emp_salary GROUP BY emp_salary)
```

2.2、求员工工资的平均数

```
-- 求员工工资的平均数
SELECT SUM(emp_salary)/COUNT(emp_name) AS avg_salary
FROM emp_salary
```

```
-- 中位数定义：将数从小到大排列，若总数为奇数，取中间位置的数值。若总数为偶数，取中间位置两个数
的平均值

-- 排位次
select emp_salary,
                row_number() over(order by emp_salary) as rn,
                count(*) over() as n
from emp_salary

--
select avg(emp_salary)
from
(
        select emp_salary,
                        row_number() over(order by emp_salary) as rn,
                        count(*) over() as n
        from emp_salary
) t
where rn in (floor(n/2)+1,if(mod(n,2) = 0,floor(n/2),floor(n/2)+1))
```

3-连续天数

问题：请求出sales_record表中连续三天有销售记录的店铺

```
create table sales_record( shopid varchar(11),dt date,sale int );

INSERT INTO sales_record
VALUES ('A', '2017-10-11', 300),
    ('A', '2017-10-12', 200),
    ('A', '2017-10-13', 100),
    ('A', '2017-10-15', 100),
    ('A', '2017-10-16', 300),
    ('A', '2017-10-17', 150),
    ('A', '2017-10-18', 340),
    ('A', '2017-10-19', 360),
    ('B', '2017-10-11', 400),
    ('B', '2017-10-12', 200),
    ('B', '2017-10-15', 600),
    ('C', '2017-10-11', 350),
    ('C', '2017-10-13', 250),
    ('C', '2017-10-14', 300),
    ('C', '2017-10-15', 400),
    ('C', '2017-10-16', 200),
    ('D', '2017-10-13', 500),
    ('E', '2017-10-14', 600),
    ('E', '2017-10-15', 500),
    ('D', '2017-10-14', 600);
```

参考答案

```
 # 解法1
# join2次
select distinct t1.shopid
from sales_record t1
left join sales_record t2
    on t1.shopid = t2.shopid
left join sales_record t3
    on t2.shopid = t3.shopid
where t1.dt = t2.dt - 1 and t2.dt = t3.dt - 1


# 解法2
 # 通过lead偏移
 SELECT DISTINCT shopid1
 FROM (
     SELECT t1.shopid AS shopid1, t1.dt AS dt1, t1.sale AS sale1
         , lead(t1.dt, 1, NULL) OVER (PARTITION BY t1.shopid ORDER BY dt) AS dt2
         , lead(t1.dt, 2, NULL) OVER (PARTITION BY t1.shopid ORDER BY dt) AS dt3
     FROM sales_record t1
 ) t2
 WHERE dt1 = dt2 - 1 AND dt2 = dt3 - 1

# 解法3
 '''给每个用户一个编号,用日期减去编号,如果是同一天,那么就是连续的'''
 # 打编号
 SELECT
```

```
shopid,
dt,
sale,
row_number() OVER (PARTITION BY shopid ORDER BY dt) AS rn
FROM sales_record;

# 根据编号生成连续日期，再进行分组求和
SELECT shopid, COUNT(1) AS count
FROM (
    SELECT
        shopid,
        dt,
        sale,
        rn,
    date_sub(dt, INTERVAL rn DAY) AS dt_sub
    FROM (
        SELECT
                shopid,
                dt,
                sale,
                row_number() OVER (PARTITION BY shopid ORDER BY dt) AS rn
        FROM sales_record
                    ) s
) s1
GROUP BY shopid, dt_sub;


# 筛选天数大于三天的并去重
SELECT DISTINCT shopid
FROM(
            SELECT shopid, COUNT(1) AS count
            FROM (
                    SELECT
                    shopid,
                    dt,
                    sale,
                    rn,
                    date_sub(dt, INTERVAL rn DAY) AS dt_sub
                    FROM (
                            SELECT
                            shopid,
                            dt,
                            sale,
                            row_number() OVER (PARTITION BY shopid ORDER BY dt)
AS rn
                            FROM sales_record
                               ) s
             ) s1
             GROUP BY shopid, dt_sub
            ) s2
 WHERE s2.count >= 3;


 # 解法4 mysql(5.7版本)
 SELECT
 shopid,
 dt,
 sale,
```

```
 row_number() OVER (PARTITION BY shopid ORDER BY dt) AS rn
 FROM sales_record;

select
shopid,
dt,
sale,
@rn:=if(@shopid=shopid,@rn:=@rn+1,@rn:=1) as rn,
@shopid:=shopid as shopid_
from sales_record s,(select @shopid:=0,@rn:=0) t


# 根据编号生成连续日期，再进行分组求和
 SELECT shopid, COUNT(1) AS count
 FROM (
     SELECT
         shopid,
         dt,
         sale,
         rn,
     date_sub(dt, INTERVAL rn DAY) AS dt_sub
     FROM (
        select
                shopid,
                dt,
                sale,
                @rn:=if(@shopid=shopid,@rn:=@rn+1,@rn:=1) as rn,
                @shopid:=shopid as shopid_
                from sales_record s,(select @shopid:=0,@rn:=0) t
                ) s
) s1
 GROUP BY shopid, dt_sub;


 # 筛选天数大于三天的并去重
 SELECT DISTINCT shopid
 FROM(
        SELECT shopid, COUNT(1) AS count
         FROM (
                SELECT
                shopid,
                dt,
                sale,
                rn,
                date_sub(dt, INTERVAL rn DAY) AS dt_sub
                FROM (
                       select
                       shopid,
                       dt,
                       sale,
                       @rn:=if(@shopid=shopid,@rn:=@rn+1,@rn:=1) as rn,
                       @shopid:=shopid as shopid_
                       from sales_record s,(select @shopid:=0,@rn:=0) t
                       ) s
        ) s1
         GROUP BY shopid, dt_sub
            ) s2
 WHERE s2.count >= 3;
```

4-连续打卡

现在有一张表t，这张表存储了每个员工每天的打卡情况

现在需要统计截止目前每个员工的连续打卡天数，表t如下表所示：

uid	tdate	is_flag
1	2020/2/1	1
1	2020/2/2	0
1	2020/2/3	1
1	2020/2/4	1
1	2020/2/5	0
1	2020/2/6	1
1	2020/2/7	1
1	2020/2/8	1
2	2020/2/1	1
2	2020/2/2	0
2	2020/2/3	0
2	2020/2/4	1
2	2020/2/5	1
2	2020/2/6	1
2	2020/2/7	1
2	2020/2/8	1

上表中uid是用户id，tdate是日期，is_flag是记录用户当天是否打卡，1为打卡，0为未打卡。

我们希望得到的结果为：

uid	flag_days
1	3
2	5

数据准备：

```
--创建 t 表:
CREATE TABLE `t`  (
  `uid` varchar(255),
  `tdate` varchar(255),
```

```
  `is_flag` varchar(255)
);
--插入数据：
INSERT INTO `t` VALUES ('1', '2020-02-01', '1');
INSERT INTO `t` VALUES ('1', '2020-02-02', '0');
INSERT INTO `t` VALUES ('1', '2020-02-03', '1');
INSERT INTO `t` VALUES ('1', '2020-02-04', '1');
INSERT INTO `t` VALUES ('1', '2020-02-05', '0');
INSERT INTO `t` VALUES ('1', '2020-02-06', '1');
INSERT INTO `t` VALUES ('1', '2020-02-07', '1');
INSERT INTO `t` VALUES ('1', '2020-02-08', '1');
INSERT INTO `t` VALUES ('2', '2020-02-01', '1');
INSERT INTO `t` VALUES ('2', '2020-02-02', '0');
INSERT INTO `t` VALUES ('2', '2020-02-03', '0');
INSERT INTO `t` VALUES ('2', '2020-02-04', '1');
INSERT INTO `t` VALUES ('2', '2020-02-05', '1');
INSERT INTO `t` VALUES ('2', '2020-02-06', '1');
INSERT INTO `t` VALUES ('2', '2020-02-07', '1');
INSERT INTO `t` VALUES ('2', '2020-02-08', '1');
```


参考答案

```
# 打编号
select
    uid,
    tdate,
    row_number() over(partition by uid order by tdate) rn
from t
where is_flag=1

# 根据编号生成连续日期，再进行分组求和
select
    uid,
    tdate,
    rn,
    date_sub(tdate,interval rn day) dt_sub
from(
        select
                uid,
                tdate,
                row_number() over(partition by uid order by tdate) rn
        from t
        where is_flag=1
    ) t1

# 分组求连续天数

select
uid,count(1)
from(
               select
                    uid,
                    tdate,
                    rn,
                    date_sub(tdate,interval rn day) dt_sub
            from(
                    select
```

```
                    uid,
                    tdate,
                    row_number() over(partition by uid order by tdate)
rn
                from t
                where is_flag=1
                ) t1
            ) t2
group by uid,dt_sub

# 最大的连续天数
select uid,max(flag_days) flag_days
from(
 select
 uid,count(1) flag_days
 from(
               select
                    uid,
                    tdate,
                    rn,
                    date_sub(tdate,interval rn day) dt_sub
            from(
                    select
                            uid,
                            tdate,
                            row_number() over(partition by uid order by tdate)
rn
                    from t
                    where is_flag=1
                    ) t1
            ) t2
group by uid,dt_sub
) t3
group by uid
```

同类题目

1. 有一张用户消费表【user_pay】，标记每天用户是否消费（说明：该表包含所有用户所有工作日的消费记录），包含三个字段：日期【date_time】，用户id【user_id】，用户当天是否消费【is_pay】：0否1是；（mysql版本5.7）

(1) 请计算截至当前日期每个用户已经连续消费的天数（输出表仅包含当天消费的所有用户，计算其连续消费天数）

(2) 请计算每个用户历史以来最大的连续消费天数（输出表为用户消费表中所有出现过的用户，计算其历史最大连续消费天数）

5-累加问题

问题：求年累加值，总累加值

```
#建表
```

```
CREATE TABLE temp (
    DATE DATETIME,
    VALUE INT
);

#插入数据
INSERT INTO temp
VALUES ('2018/11/23', 10),
    ('2018/11/25', 12),
    ('2018/12/31', 3),
    ('2019/2/9', 53),
    ('2019/3/31', 23),
    ('2019/7/8', 11),
    ('2019/7/31', 10);
```

参考答案:

```
 -- 求年累加值，总累加值
 -- 分年月聚合value
SELECT
YEAR(DATE) AS YEAR,
MONTH(DATE) AS MONTH,
SUM(VALUE) AS VALUE
FROM temp
GROUP BY YEAR(DATE), MONTH(DATE)

-- 求年累加和总累加
SELECT
YEAR,
MONTH,
SUM(VALUE) OVER (PARTITION BY YEAR ORDER BY YEAR,MONTH ASC) AS YSUM,
SUM(VALUE) OVER(ORDER BY YEAR,MONTH ASC) CUM_SUM
FROM(
    SELECT
    YEAR(DATE) AS YEAR,
    MONTH(DATE) AS MONTH,
    SUM(VALUE) AS VALUE
    FROM temp
    GROUP BY YEAR(DATE), MONTH(DATE)
) t


 -- MYSQL5.7版本
 SET @year_sum = 0;
 SET @cum_sum = 0;
 SET @year = 0;
 SELECT b.YEAR, b.MONTH, b.YSUM, b.CUM_SUM
 FROM (
     SELECT a.*
         , @year_sum := IF(@year = YEAR, @year_sum + VALUE, VALUE) AS YSUM
         , @year := YEAR
         , @cum_sum := @cum_sum + VALUE AS CUM_SUM
     FROM (
         SELECT YEAR(DATE) AS YEAR, MONTH(DATE) AS MONTH
             , SUM(VALUE) AS VALUE
         FROM temp
         GROUP BY YEAR(DATE), MONTH(DATE)
```

```
    ) a
) b
```