# 第1题

我们有如下的用户访问数据

userId visitDate visitCount

u01 2017/1/21 5

u02 2017/1/23 6

u03 2017/1/22 8

u04 2017/1/20 3

u01 2017/1/23 6

u01 2017/2/21 8

U02 2017/1/23 6

U01 2017/2/22 4

要求使用SQL统计出每个用户的累积访问次数，如下表所示：

用户id 月份 小计 累积

u01 2017-01 11 11

u01 2017-02 12 23

u02 2017-01 12 12

u03 2017-01 8 8

u04 2017-01 3 3

1.建表语句

create table first(userid string,visitDate String,visitCount string);

2.插入数据

insert into table first values('u01','2017/1/21','5');

insert into table first values('u02','2017/1/23','6');

insert into table first values('u03','2017/1/22','8');

insert into table first values('u04','2017/1/20','3');

insert into table first values('u01','2017/1/23','6');

insert into table first values('u01','2017/2/21','8');

insert into table first values('u02','2017/1/23','6');

insert into table first values('u01','2017/2/22','4');

select

date\_format(visitDate,'yyyy-MM') aa

from first

group by aa;

3.转变时间格式

select from\_unixtime(unix\_timestamp(visitDate,'yyyy/mm/dd'),'yyyy-mm') ;

4.需求分析

首先按照月份分组，算出每个月的总访问次数

然后用窗口函数，按照月份计算累加访问次数

4.1格式化时间

select

userId,

from\_unixtime(unix\_timestamp(visitDate,'yyyy/mm/dd'),'yyyy-mm') month,

visitCount

from first; --------t1表

4.2求月份的总次数

select

userId,

month,

sum(visitCount) sum\_month

from t1

group by userId,month; ---------t2表

4.3按照用户id开窗，按时间排序

select

userId,

month,

sum\_month,

sum(sum\_month) over (partition by userID order by month rows between UNBOUNDED PRECEDING and current row) sum\_all

from t2

5.终级sql

select

t2.userId,

t2.month,

sum\_month,

sum(t2.sum\_month) over (partition by userId order by month rows between UNBOUNDED PRECEDING and current row) sum\_all

from

(

select

t1.userId userId,

t1.month month,

sum(t1.visitCount) sum\_month

from

(

select

userId,

from\_unixtime(unix\_timestamp(visitDate,'yyyy/mm/dd'),'yyyy-mm') month,

visitCount

from first

) t1

group by userId,month

)t2;

# 第2题 京东

有50W个京东店铺，每个顾客访客访问任何一个店铺的任何一个商品时都会产生一条访问日志，访问日志存储的表名为Visit，访客的用户id为user\_id，被访问的店铺名称为shop，请统计：

1）每个店铺的UV（访客数）

2）每个店铺访问次数top3的访客信息。输出店铺名称、访客id、访问次数

create table Second\_Visit (user\_id string,shop string);

insert into table second\_visit values ('1','a');

insert into table second\_visit values ('1','b');

insert into table second\_visit values ('2','a');

insert into table second\_visit values ('3','c');

insert into table second\_visit values ('1','a');

insert into table second\_visit values ('1','a');

1.每个店铺的UV（访客数）

select

t1.shop,

count(\*)

from

(

select

user\_id,

shop

from second\_visit

group by user\_id,shop

)t1

group by shop;

2.每个店铺访问次数top3的访客信息，输出店铺名称，访客id，访问次数

select

t2.shop,

t2.user\_id,

t2.num

from

(

select

t1.user\_id user\_id,

t1.shop shop,

t1.num,

rank () over (partition by t1.user\_id order by num) con

from

(

select

user\_id,

shop,

count(\*) num

from second\_visit

group by user\_id,shop

)t1

)t2

where con<=3;

# 第3题

已知一个表STG.ORDER，有如下字段:Date，Order\_id，User\_id，amount。请给出sql进行统计:数据样例:2017-01-01,10029028,1000003251,33.57。

1）给出 2017年每个月的订单数、用户数、总成交金额。

2）给出2017年11月的新客数(指在11月才有第一笔订单)

1.create table second\_order(`Date` String,Order\_id String,User\_id String,amount double);

2.--样例数据

同一个用户，相同月份

insert into table second\_order values ('2017-01-01','10029028','1000003251',33.57);

insert into table second\_order values ('2017-01-01','10029029','1000003251',33.57);

不同用户,相同月份

insert into table second\_order values ('2017-01-01','100290288','1000003252',33.57);

不同月份

insert into table second\_order values ('2017-02-02','10029088','1000003251',33.57);

insert into table second\_order values ('2017-02-02','100290281','1000003251',33.57);

insert into table second\_order values ('2017-02-02','100290282','1000003253',33.57);

insert into table second\_order values ('2017-11-02','10290282','100003253',234);

insert into table second\_order values ('2017-11-02','10290282','100003243',234);

3.需求分析

3.1

先求出订单数和总成交额为result1，然后在算出每个月的用户数result2，然后两个表做join操作，最终求出结果。

select

result1.month,

result1.count\_order,

result1.count\_amount,

result2.count\_user

from

(select

t1.month month,

count(t1.Order\_id) count\_order,

sum(t1.amount) count\_amount

from

(

select

date\_format(`Date`,'yyyy-MM') month,

Order\_id,

User\_id,

amount

from second\_order

)t1

group by month

)result1

join

(select

t2.month month,

count(\*) count\_user

from

(

select

t1.month month,

t1.Order\_id,

t1.User\_id,

row\_number() over(partition by t1.month,t1.User\_id order by amount) con

from

(

select

date\_format(`Date`,'yyyy-MM') month,

Order\_id,

User\_id,

amount

from second\_order

)t1

) t2

where con=1

group by month

)result2

on result2.month=result1.month;

简单写法：

SELECT

count(Order\_id) order\_count,

count(DISTINCT(User\_id)) user\_count,

sum(amount) amount\_sum,

substring(`Date`, 1, 7)

FROM

second\_order

WHERE

substring(`Date`, 1, 4) = '2017'

GROUP BY

substring(`Date`, 1, 7);

3.2按用户id分组，count为1，并且date是在11月份

select

count(\*)

from

(

select

user\_id

from

(

select

`user\_id`,

`date`,

row\_number() over(partition by user\_id order by `date` desc ) shop\_count

from second\_order

) t1

where date\_format(`date`, 'yyyy-MM') = '2017-11' and shop\_count = 1

) t2;

# 第4题

有一个5000万的用户文件(user\_id，name，age)，一个2亿记录的用户看电影的记录文件(user\_id，url)，根据年龄段观看电影的次数进行排序？

1.建表

create table forth\_user(user\_id string,name string,age int);

create table forth\_log(user\_id string,url string);

insert into table forth\_user values('001','wt',10);

insert into table forth\_user values('002','ls',18);

insert into table forth\_user values('003','zz',30);

insert into table forth\_user values('004','zz',50);

insert into table forth\_log values('001','sdf');

insert into table forth\_log values('001','wss');

insert into table forth\_log values('002','sdf');

insert into table forth\_log values('003','sdf');

insert into table forth\_log values('004','sdf');

2.分析需求

先求出每个人看了几次电影,t1

然后t1和user表join，拼接age字段 t2表

划分年龄段，0-20，20-40，40-60，60--

按年龄段分组，按照次数排序

select

user\_id,

count(\*) con

from forth\_log

group by user\_id; ----t1

select

u1.age age ,

t1.user\_id,

t1.con con

from forth\_user u1

join

(

select

user\_id,

count(\*) con

from forth\_log

group by user\_id

)t1

on u1.user\_id=t1.user\_id --t2

select

t2.con con,

case

when 0<=t2.age and t2.age<20 then 'a'

when 20<=t2.age and t2.age<40 then 'b'

when 40<=t2.age and t2.age<60 then 'c'

else 'd'

end as category

from t2 ---t3

select

t3.category

sum(t3.con)

from t3

gruop by t3.category

=====================================================================

3.终极sql

select

t4.category,

t4.sumcon

from

(

select

t3.category category,

sum(t3.con) sumcon

from

(

select

t2.con con,

case

when 0<=t2.age and t2.age<20 then 'a'

when 20<=t2.age and t2.age<40 then 'b'

when 40<=t2.age and t2.age<60 then 'c'

else 'd'

end as category

from

(

select

u1.age,

t1.user\_id,

t1.con con

from forth\_user u1

join

(

select

user\_id,

count(\*) con

from forth\_log

group by user\_id

)t1

on u1.user\_id=t1.user\_id

)t2

)t3

group by t3.category

)t4

order by t4.sumcon

# 第5题

有日志如下，请写出代码求得所有用户和活跃用户的总数及平均年龄。（活跃用户指连续两天都有访问记录的用户）

日期 用户 年龄

11,test\_1,23

11,test\_2,19

11,test\_3,39

11,test\_1,23

11,test\_3,39

11,test\_1,23

12,test\_2,19

13,test\_1,23

create table fiveth(`date` string,user\_id string ,age int );

insert into table fiveth values ('11','test\_1',23);

insert into table fiveth values ('11','test\_2',19);

insert into table fiveth values ('11','test\_3',39);

insert into table fiveth values ('11','test\_1',23);

insert into table fiveth values ('11','test\_3',39);

insert into table fiveth values ('11','test\_1',23);

insert into table fiveth values ('12','test\_2',19);

insert into table fiveth values ('13','test\_1',23);

总人数和平均年龄

1.每个人的年龄相同，按照user\_id，和age分组

select

count(\*),

avg(age)

from

(

select

user\_id,age

from

fiveth

group by user\_id,age

)t1

活跃人数和平均年龄

1.先按照日期和用户去重

2.开窗函数，利用等差数列

select

user\_id,`date`,age

from

fiveth group by

user\_id,`date`,age --fiveth1

select

`date`

user\_id,

age,

rank() over(partition by user\_id order by `date`) rank

from

fiveth1 --t1

select

t1.`date`-rank `date\_dif`,

user\_id,

age

from t1 ---t2

select

t2.age,

t2.user\_id

from t2

group by t2.user\_id,t2.`date\_fid`,t2.age

having count(\*) >=2; --t3

select

count(\*),

avg(t3.age)

from t3

终极sql

select

count(\*),

avg(t3.age)

from

(

select

t2.age,

t2.user\_id

from

(

select

t1.`date`- rank `date\_dif`,

user\_id,

age

from

(

select

`date`,

user\_id,

age,

rank() over(partition by user\_id order by `date`) rank

from

(

select

user\_id,`date`,age

from

fiveth

group by

user\_id,`date`,age

)fiveth1

)t1

)t2

group by t2.user\_id,t2.`date\_dif`,t2.age

having count(\*) >=2

)t3

# 第6题

请用sql写出所有用户中在今年10月份第一次购买商品的金额，表ordertable字段

（购买用户：userid，金额：money，购买时间：paymenttime(格式：2017-10-01)，订单id：orderid）

create table sixth (userid string,monty string ,paymenttime string,orderid string);

insert into table sixth values('001','100','2017-10-01','123123');

insert into table sixth values('001','200','2017-10-02','123124');

insert into table sixth values('002','500','2017-10-01','222222');

insert into table sixth values('001','100','2017-11-01','123123');

选出所有用户十月份的购买记录

然后选出每个人在十月份第一条购买记录的金额

select

userid,

paymenttime,

monty

from

(

select

paymenttime,

userid,

monty,

orderid,

row\_number() over(partition by userid order by paymenttime) row\_con

from sixth

where date\_format(paymenttime,'yyyy-MM')='2017-10'

) t1

where t1.row\_con=1;

# 第7题

现有图书管理数据库的三个数据模型如下：

图书（数据表名：BOOK）

序号 字段名称 字段描述 字段类型

1 BOOK\_ID 总编号 文本

2 SORT 分类号 文本

3 BOOK\_NAME 书名 文本

4 WRITER 作者 文本

5 OUTPUT 出版单位 文本

6 PRICE 单价 数值（保留小数点后2位）

读者（数据表名：READER）

序号 字段名称 字段描述 字段类型

1 READER\_ID 借书证号 文本

2 COMPANY 单位 文本

3 NAME 姓名 文本

4 SEX 性别 文本

5 GRADE 职称 文本

6 ADDR 地址 文本

借阅记录（数据表名：BORROW LOG）

序号 字段名称 字段描述 字段类型

1 READER\_ID 借书证号 文本

2 BOOK\_ID 总编号 文本

3 BORROW\_DATE 借书日期 日期

（1）创建图书管理库的图书、读者和借阅三个基本表的表结构。请写出建表语句。

（2）找出姓李的读者姓名（NAME）和所在单位（COMPANY）。

（3）查找“高等教育出版社”的所有图书名称（BOOK\_NAME）及单价（PRICE），结果按单价降序排序。

（4）查找价格介于10元和20元之间的图书种类(SORT）出版单位（OUTPUT）和单价（PRICE），结果按出版单位（OUTPUT）和单价（PRICE）升序排序。

（5）查找所有借了书的读者的姓名（NAME）及所在单位（COMPANY）。

（6）求”科学出版社”图书的最高单价、最低单价、平均单价。

（7）找出当前至少借阅了2本图书（大于等于2本）的读者姓名及其所在单位。

（8）考虑到数据安全的需要，需定时将“借阅记录”中数据进行备份，请使用一条SQL语句，在备份用户bak下创建与“借阅记录”表结构完全一致的数据表BORROW\_LOG\_BAK.井且将“借阅记录”中现有数据全部复制到BORROW\_L0G\_ BAK中。

（9）现在需要将原Oracle数据库中数据迁移至Hive仓库，请写出“图书”在Hive中的建表语句（Hive实现，提示：列分隔符|；数据表数据需要外部导入：分区分别以month＿part、day＿part 命名）

（10）Hive中有表A，现在需要将表A的月分区　201505　中　user＿id为20000的user＿dinner字段更新为bonc8920，其他用户user＿dinner字段数据不变，请列出更新的方法步骤。（Hive实现，提示：Hlive中无update语法，请通过其他办法进行数据更新）

1.创建图书表book

create table book(book\_id string,sort string,book\_name string,writer string,output string,price decimal(10,2));

创建读者表reader

create table reader (reader\_id string,company string,name string,sex string ,grade string,addr string);

insert into table reader values ('001','sgg','lisi','man','1','beijing');

insert into table reader values ('002','tencent','wt','man','2','shanghai');

创建借阅记录表borrow\_log

create table borrow\_log(reader\_id string,book\_id string ,borrow\_date string);

2.

select

name ,

company

from

reader

where name like 'li%';

3. select

book\_name,

price

from book

where output='高等教育出版社'

order by price desc;

4. select

sort,

output,

price

from book

where price <=20 and price >=10

order by output,price ;

5. select

t1.name,

t1.company

from

(

select

r.name name,

r.company company

from borrow\_log b

join reader r on

b.reader\_id=r.reader\_id

) t1

group by t1.name,t1.company;

6. select

max(price),

min(price),

avg(price)

from book

where output ='科学出版社';

7. select

t1.name,

t1.company

from

(

select

r.name name,

r.company company

from borrow\_log b

join reader r on

b.reader\_id=r.reader\_id

) t1

group by t1.name,t1.company having count(\*)>=2;

8. create table if not exists borrow\_log\_bak

select \* from borrow\_log;

9. create table book\_hive(book\_id string,sort string,book\_name string,writer string,output string,price decimal(10,2))

partitioned by (month\_part string,day\_part string)

row format delimited fields terminated by '\\|'

stored as textfile;

10. hive在1.1.0版本之前不可以更新数据，在之后可以更改

同样在建表后面添加: stored as orc TBLPROPERTIES('transactional'='true')

但update操作非常慢

# 第8题

有一个线上服务器访问日志格式如下（用sql答题）

时间 接口 ip地址

2016-11-09 11:22:05 /api/user/login 110.23.5.33

2016-11-09 11:23:10 /api/user/detail 57.3.2.16

2016-11-09 23:59:40 /api/user/login 200.6.5.166

求11月9号下午14点（14-15点），访问api/user/login接口的top10的ip地址

create table eight\_log(`date` string,interface string ,ip string);

insert into table eight\_log values ('2016-11-09 11:22:05','/api/user/login','110.23.5.23');

insert into table eight\_log values ('2016-11-09 11:23:10','/api/user/detail','57.3.2.16');

insert into table eight\_log values ('2016-11-09 23:59:40','/api/user/login','200.6.5.166');

insert into table eight\_log values('2016-11-09 11:14:23','/api/user/login','136.79.47.70');

insert into table eight\_log values('2016-11-09 11:15:23','/api/user/detail','94.144.143.141');

insert into table eight\_log values('2016-11-09 11:16:23','/api/user/login','197.161.8.206');

insert into table eight\_log values('2016-11-09 12:14:23','/api/user/detail','240.227.107.145');

insert into table eight\_log values('2016-11-09 13:14:23','/api/user/login','79.130.122.205');

insert into table eight\_log values('2016-11-09 14:14:23','/api/user/detail','65.228.251.189');

insert into table eight\_log values('2016-11-09 14:15:23','/api/user/detail','245.23.122.44');

insert into table eight\_log values('2016-11-09 14:17:23','/api/user/detail','22.74.142.137');

insert into table eight\_log values('2016-11-09 14:19:23','/api/user/detail','54.93.212.87');

insert into table eight\_log values('2016-11-09 14:20:23','/api/user/detail','218.15.167.248');

insert into table eight\_log values('2016-11-09 14:24:23','/api/user/detail','20.117.19.75');

insert into table eight\_log values('2016-11-09 15:14:23','/api/user/login','183.162.66.97');

insert into table eight\_log values('2016-11-09 16:14:23','/api/user/login','108.181.245.147');

insert into table eight\_log values('2016-11-09 14:17:23','/api/user/login','22.74.142.137');

insert into table eight\_log values('2016-11-09 14:19:23','/api/user/login','22.74.142.137');

select

t1.ip,

t1.con

from

(

select

ip,

count(\*) con

from eight\_log

where date\_format(`date`,'yyyy-MM-dd HH')='2016-11-09 14' and interface ='/api/user/login'

group by ip

)t1

order by con desc limit 10;

# 第9题

有一个充值日志表如下：

CREATE TABLE `credit log`

(

`dist\_id` int（11）DEFAULT NULL COMMENT '区组id',

`account` varchar（100）DEFAULT NULL COMMENT '账号',

`money` int(11) DEFAULT NULL COMMENT '充值金额',

`create\_time` datetime DEFAULT NULL COMMENT '订单时间'

)ENGINE=InnoDB DEFAUILT CHARSET-utf8

请写出SQL语句，查询充值日志表2015年7月9号每个区组下充值额最大的账号，要求结果：

区组id，账号，金额，充值时间

create table nine\_log(

dist\_id int,

account string,

money int,

create\_time string

)

insert into table nine\_log values (1,'001',100,'2015-07-09');

insert into table nine\_log values (1,'002',500,'2015-07-09');

insert into table nine\_log values (2,'001',200,'2015-07-09');

select

t1.dist\_id,

t1.account,

t1.money,

t1.create\_time

from

(

select

dist\_id,

account,

create\_time,

money,

rank() over(partition by dist\_id order by money desc) rank

from nine\_log

where create\_time='2015-07-09'

)t1

where rank=1;

# 第10题

有一个账号表如下，请写出SQL语句，查询各自区组的money排名前十的账号（分组取前10）

CREATE TABIE `account`

(

`dist\_id` int（11）DEFAULT NULL COMMENT '区组id'，

`account` varchar（100）DEFAULT NULL COMMENT '账号' ,

`gold` int（11）DEFAULT NULL COMMENT '金币'

PRIMARY KEY （`dist\_id`，`account\_id`），

）ENGINE=InnoDB DEFAULT CHARSET-utf8

create table ten\_log(

dist\_id int,

account string,

money int

)

insert into table ten\_log values (1,'001',100);

insert into table ten\_log values (1,'002',500);

insert into table ten\_log values (2,'001',200);

--------------------mysql写法------------------

不会

-----------------hive的写法---------------------

select

t1.dist\_id,

t1.account,

t1.money

from

(

select

dist\_id,

account,

money,

rank() over (partition by dist\_id order by money desc ) rank

from ten\_log

)t1

where t1.rank <=10;

# 第11题

1）有三张表分别为会员表（member）销售表（sale）退货表（regoods）

（1）会员表有字段memberid（会员id，主键）credits（积分）；

（2）销售表有字段memberid（会员id，外键）购买金额（MNAccount）；

（3）退货表中有字段memberid（会员id，外键）退货金额（RMNAccount）；

2）业务说明：

（1）销售表中的销售记录可以是会员购买，也可是非会员购买。（即销售表中的memberid可以为空）

（2）销售表中的一个会员可以有多条购买记录

（3）退货表中的退货记录可以是会员，也可是非会员4、一个会员可以有一条或多条退货记录

查询需求：分组查出销售表中所有会员购买金额，同时分组查出退货表中所有会员的退货金额，

把会员id相同的购买金额-退款金额得到的结果更新到表会员表中对应会员的积分字段（credits）

1. select

s.memberid memberid,

s.mnaccount mnaccount,

r.rmnaccount rmnaccount

from sale s

join regoods r

on s.memberid=r.memberid --------t1

select

t1.memberid,

sum(t1.mnaccount) sum\_buy,

sum(t1.rmnaccount) sum\_tui,

sum\_buy-sum\_tui sum\_dif

from t1

group by t1.memberid -----t2

insert into member select t2.memberid,t2.sum\_dif from t2 ;

# 第12题 百度

现在有三个表student（学生表）、course(课程表)、score（成绩单），结构如下：

create table student

(

id bigint comment ‘学号’，

name string comment ‘姓名’,

age bigint comment ‘年龄’

);

create table course

(

cid string comment ‘课程号，001/002格式’,

cname string comment ‘课程名’

);

Create table score

(

Id bigint comment ‘学号’,

cid string comment ‘课程号’,

score bigint comment ‘成绩’

) partitioned by(event\_day string)

其中score中的id、cid，分别是student、course中对应的列请根据上面的表结构，回答下面的问题

1）请将本地文件（/home/users/test/20190301.csv）文件，加载到分区表score的20190301分区中，并覆盖之前的数据

2）查出平均成绩大于60分的学生的姓名、年龄、平均成绩

3）查出没有‘001’课程成绩的学生的姓名、年龄

4）查出有‘001’\’002’这两门课程下，成绩排名前3的学生的姓名、年龄

5）创建新的表score\_20190317，并存入score表中20190317分区的数据

6）如果上面的score表中，uid存在数据倾斜，请进行优化，查出在20190101-20190317中，学生的姓名、年龄、课程、课程的平均成绩

7）描述一下union和union all的区别，以及在mysql和HQL中用法的不同之处？

8）简单描述一下lateral view语法在HQL中的应用场景，并写一个HQL实例

1.load data local inpath '/home/users/test/20190301.csv' overwrite into table score partition (event\_day='20190301');

2.

select

id,

s.name,

s.age,

avg(score)

from score

group by id having avg(score)>60

join

student s

on s.id=score.id

3.

4.

5.create table if not exists score\_20190317 as select \* from score where event\_dayk='20190317';

6.

select

uid,

cid,

avg(score),

s.age

from score

where event\_day>='20190101' and event\_day<='20190317'

group by uid,cid

join

student s

on s.id=score.uid.

7.union 会对结果进行去重

8.lateral view 和udtf函数一起使用，用于将一列炸裂为很多行

select

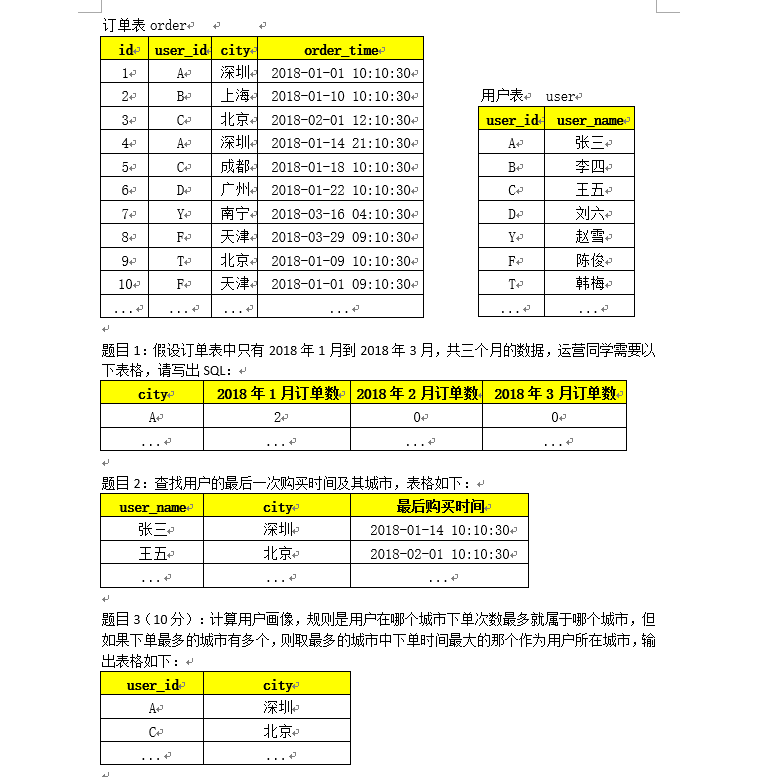
movie,

category\_name

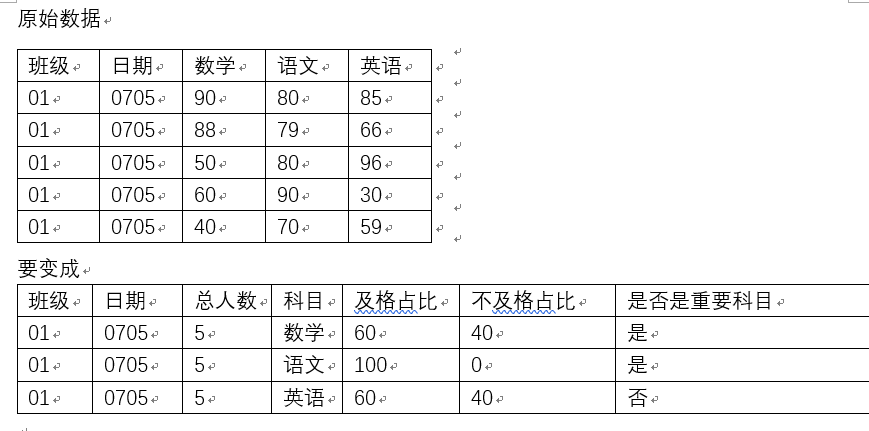
from movie\_info

lateral view explode(category) table\_temp as category\_name;

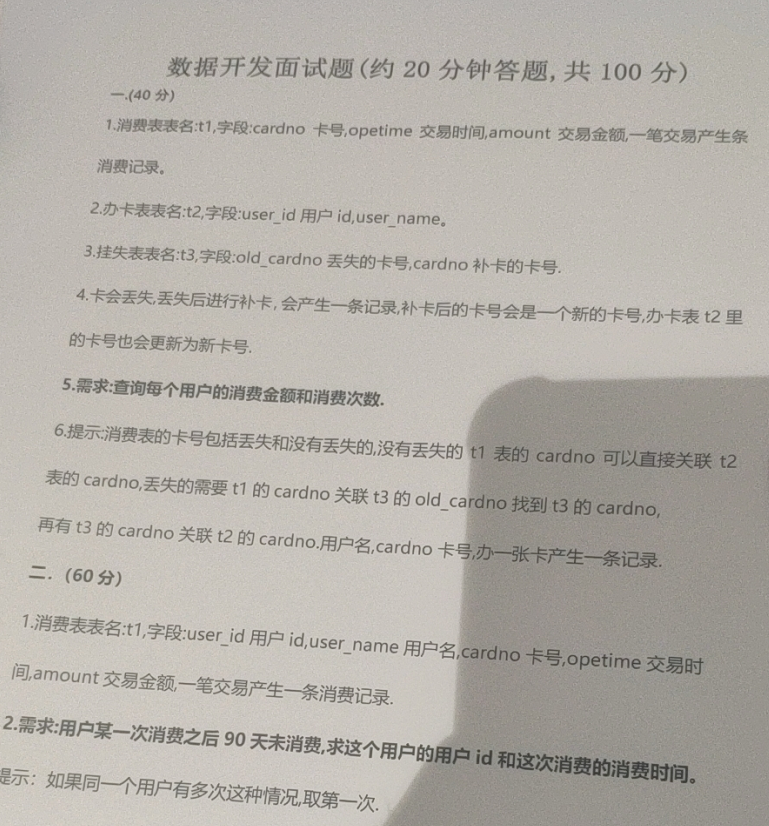
# 第13题



# 第14题



# 第15题



1：查询每个用户的消费金额和消费次数：

如果没有丢失的话：

t1关联t2即可

select sum(t1.amount) as everyMout, count(t1.amount) from t1 join t2 on t1.cardno = t2.cardno group by t2.userid;

如果存在挂失表，需要获取到旧有卡号

先查询出每张表的旧有卡号：

select

sum(t1.amount) as everyMout, count(t1.amount)

from

(select

(case when t3.old\_cardno is null then t2.cardno

else t3.old\_cardno end ) as cardno,t2.amount as amount

from t2 left join t3 on t2.cardno =t3.old\_cardno )

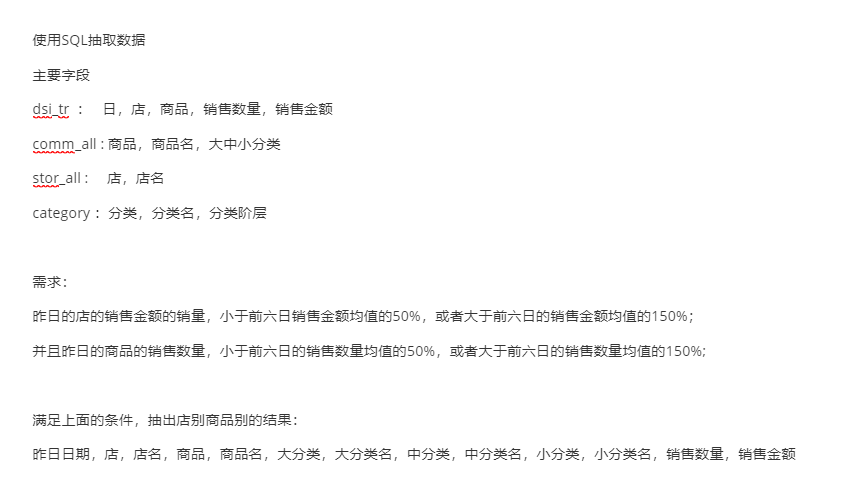
t

join

t2

on t.cardno = t2.cardno group by t2.userid;

# 第16题



第一大步：计算出基础数据

第一步，

计算出昨日商店的销售额，写入到shop\_trade、

商品的销售数量 写入到googds\_trade、

第二步，

计算出商店前六日的销售金额均值、写入到 shop\_last6\_trade

商品的销售数量均值 写入到 goods\_last6\_trade

第二大步：查询出满足条件的商店与商品

第三步，shop\_trade 与 shop\_last6\_trade 内连接，查询出满足条件的商店，数据写入到 tmp\_shop\_trade

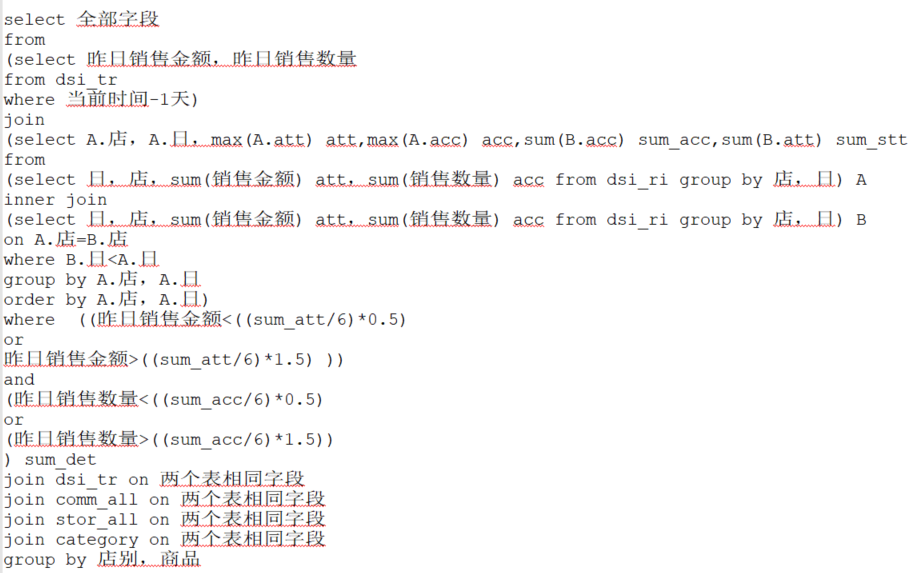
第四步，googds\_trade 与 goods\_last6\_trade 内连接，查询出满足条件的商店，数据写入到 tmp\_goods\_trade

第三大步：汇总获取数据

第五步：tmp\_shop\_trade 与 tmp\_goods\_trade 内连接，抽出商品，写入到tmp\_shop\_goods\_trade

第四大步：组装报表

第六步：tmp\_shop\_goods\_trade分别于 comm\_all,stor\_all,category等维度表关联，组装成需要的报表



select 全部字段

from

(select 昨日销售金额，昨日销售数量

from dsi\_tr

where 当前时间-1天)

join

(select A.店，A.日，max(A.att) att,max(A.acc) acc,sum(B.acc) sum\_acc,sum(B.att) sum\_stt

from

(select 日，店，sum(销售金额) att，sum(销售数量) acc from dsi\_ri group by 店，日) A

inner join

(select 日，店，sum(销售金额) att，sum(销售数量) acc from dsi\_ri group by 店，日) B

on A.店=B.店

where B.日<A.日

group by A.店，A.日

order by A.店，A.日)

where ((昨日销售金额<((sum\_att/6)\*0.5)

or

昨日销售金额>((sum\_att/6)\*1.5) ))

and

(昨日销售数量<((sum\_acc/6)\*0.5)

or

(昨日销售数量>((sum\_acc/6)\*1.5))

) sum\_det

join dsi\_tr on 两个表相同字段

join comm\_all on 两个表相同字段

join stor\_all on 两个表相同字段

join category on 两个表相同字段

group by 店别，商品