千锋好程序员大数据之场景真题锦集

制作人	版本	备注
李亚东	v1.0	hive场景题更新到25到题，所有题都重要。
李亚东	v4.0	hive场景题更新到35道题。 注:所有题均为重点,难度均为中等难度,要求所有人必须会。
李亚东|曹庆海	V5.0	hive场景题更新到49道题。

简介:

为提高学员面试中hive场景题的handle能力，本场景真题锦集全部来自于千锋往期大数据学员，适合于有大数据基础人员及在职人员阅读，希望各位勿喷。

本锦集并无统一标准答案,千锋教学部仅提供参考答案，以供大家参考，如大家有更好参考答案，可整理、讨论并更新。参考答案有不合适之处，希望大家能理解并及时联系千锋大数据相关人员及学员，我们会第一时间去更新。

最后非常感谢参与该文档编写、更新、矫正等贡献的所有老师和同学！！！

目录

1 hive场景题

1.1 hive窗口函数

1.1.1 了解哪些窗口函数，都是什么功能？找一个在某个业务中的应用？

参考答案:

```
sum(col) over() :  分组对col累计求和，over() 中的语法如下
count(col) over() : 分组对col累计，over() 中的语法如下
min(col) over() : 分组对col求最小
```

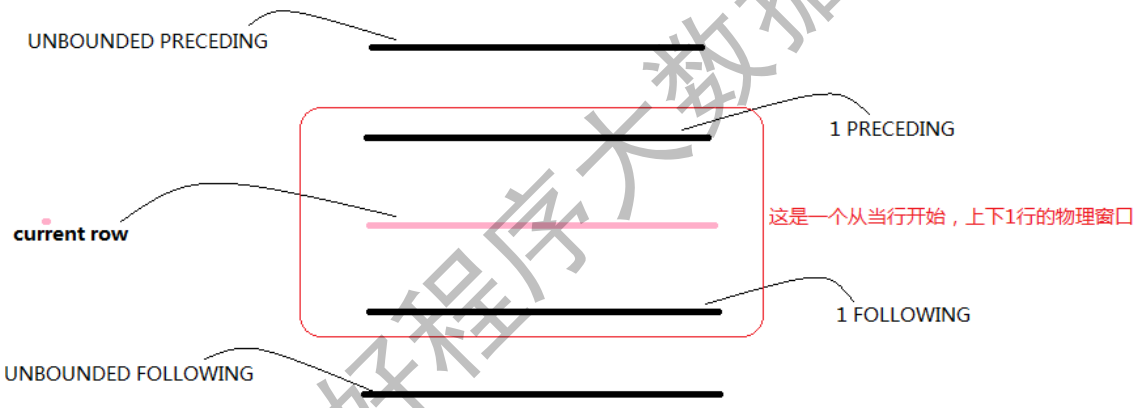
max(col) over() : 分组求col的最大值
avg(col) over() : 分组求col列的平均值
first_value(col) over() : 某分区排序后的第一个col值
last_value(col) over() : 某分区排序后的最后一个col值
lag(col,n,DEFAULT) : 统计往前n行的col值，n可选，默认为1，DEFAULT当往上第n行为NULL时候，取默认值，如不指定，则为NULL
lead(col,n,DEFAULT) : 统计往后n行的col值，n可选，默认为1，DEFAULT当往下第n行为NULL时候，取默认值，如不指定，则为NULL
ntile(n) : 用于将分组数据按照顺序切分成n片，返回当前切片值。注意：n必须为int类型。

排名函数：
row_number() over() : 排名函数，不会重复，适合于生成主键或者不并列排名
rank() over() : 排名函数，有并列名次，名次不连续。如:1,1,3
dense_rank() over() : 排名函数，有并列名次，名次连续。如：1，1，2

over(分组 排序 窗口) 中的order by后的语法:
1、物理窗口（真实往上下移动多少行rows between）：

CURRENT ROW | UNBOUNDED PRECEDING | [num] PRECEDING AND UNBOUNDED FOLLOWING | [num] FOLLOWING| CURRENT ROW

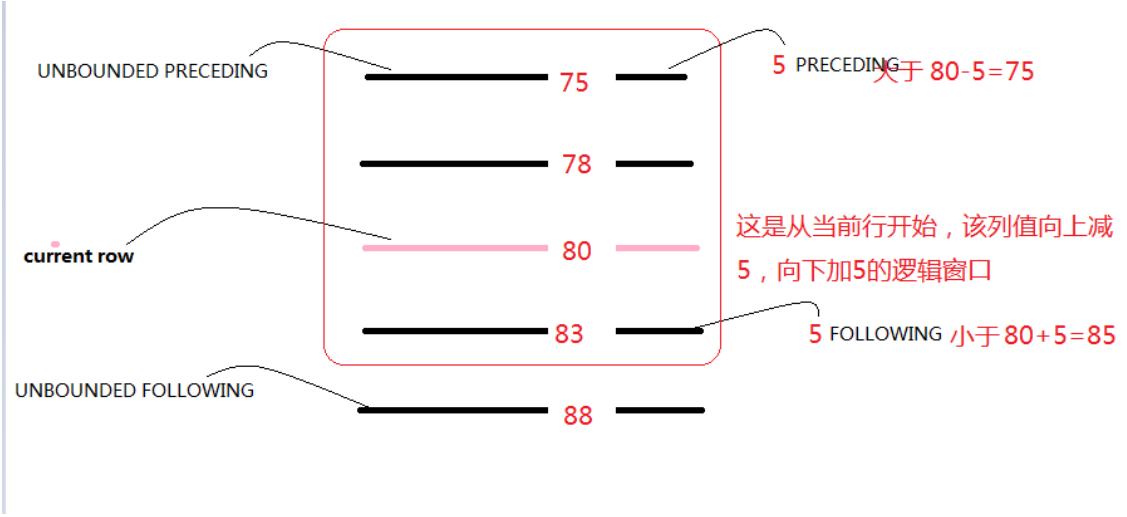
如: over(partition by col order by rows between 1 preceding and 1 fllowing)



2、 逻辑窗口(满足条件上下多少行):

range between [num] PRECEDING AND [num] FOLLOWING

如: over(partition by col order by range between 5 preceding and 5 fllowing)



注意:窗口函数一般不和group by搭配使用。

应用:
某天某产品的累计销售额。

1.1.2 编写sql实现每个用户截止到每月为止的最大单月访问次数和累计到该月的总访问次数

数据:

```
userid,month,visits
A,2015-01,5
A,2015-01,15
B,2015-01,5
A,2015-01,8
B,2015-01,25
A,2015-01,5
A,2015-02,4
A,2015-02,6
B,2015-02,10
B,2015-02,5
A,2015-03,16
A,2015-03,22
B,2015-03,23
B,2015-03,10
B,2015-03,1
```

每个用户截止到每月为止的最大单月访问次数和累计到该月的总访问次数，结果数据格式如下:

用户	月份	最大访问次数	总访问次数	当月访问次数
A	2015-01	33	33	33
A	2015-02	33	43	10
A	2015-03	38	81	38
B	2015-01	30	30	30
B	2015-02	30	45	15
B	2015-03	44	89	44

参考答案:

```
create table visits(
userid int,
month string,
visits int
)
row format delimited fields terminated by ','
;

load data local inpath '/hivedata/visits.txt' overwrite into table visits;

select
userid,
month,
visits,
max(visits) over(distribute by userid sort by month) maxvisit,
sum(visits) over(distribute by userid sort by month) totalvisit
from
(select
userid,
month,
sum(visits) visits
from visits
group by userid,month) t1
;
```

1.1.3 求出每个栏目的被观看次数及累计观看时长

```
数据:
vedio表
Uid channl min
1   1   23
2   1   12
3   1   12
4   1   32
5   1   342
6   2   13
7   2   34
8   2   13
9   2   134
```

参考答案:

```
参考答案:
create table video(
Uid int,
channel string,
min int
)
row format delimited
fields terminated by ' '
;
load data local inpath '/hivedata/video.txt' into table video;

select
```

```
channel,
count(1) num,
sum(min) total
from video
group by channel
;
```

1.1.4 编写连续7天登录的总人数

```
数据:
t1表
Uid  dt  login_status(1登录成功,0异常)
1  2019-07-11 1
1  2019-07-12 1
1  2019-07-13 1
1  2019-07-14 1
1  2019-07-15 1
1  2019-07-16 1
1  2019-07-17 1
1  2019-07-18 1
2  2019-07-11 1
2  2019-07-12 1
2  2019-07-13 0
2  2019-07-14 1
2  2019-07-15 1
2  2019-07-16 0
2  2019-07-17 1
2  2019-07-18 0
3  2019-07-11 1
3  2019-07-12 1
3  2019-07-13 1
3  2019-07-14 1
3  2019-07-15 1
3  2019-07-16 1
3  2019-07-17 1
3  2019-07-18 1
```


参考答案:

```
create table login(
Uid int,
dt string,
login_status int
)
row format delimited fields terminated by ' '
;

load data local inpath '/hivedata/login.txt' into table login;


select
uid,
dt
from
```

```
(select
t1.uid uid,
date_sub(t1.dt,t1.rm) dt
from
(select
uid,
dt,
row_number() over(distribute by uid sort by dt) rm
from login
where
login_status=1) t1)t2
group by uid,dt
having count(uid) >7
;
```

1.1.5 你知道的排名函数有哪些？说一说它们之间的区别？

参考答案:

```
排名函数：
row_number() over() ： 排名函数，没有并列名次，名次连续，如:1,2,3. 适合于生成主键或者不并列排名，
rank() over() ：  排名函数，有并列名次，名次不连续。如:1,1,3.
dense_rank() over() ： 排名函数，有并列名次，名次连续。如：1,1,2.

区别：
1、(共同点)三者都可以用于分组排名，名次都是递增。
2、(不同点)row_number()没有并列名次，名次连续，如:1,2,3; rank()有并列名次，名次不连续，如:1,1,3。dense_rank()有并列名次，名次连续。如：1,1,2。
```

1.1.6 编写sql语句实现每班前三名，分数一样并列，同时求出前三名按名次排序的一次的分差：

```
数据：
stu表
Stu_no class score
1   1901    90
2   1901    90
3   1901    83
4   1901    60
5   1902    66
6   1902    23
7   1902    99
8   1902    67
9   1902    87
```

编写sql实现，结果如下：

```
结果数据：
班级  stu_no  score   rn rn1  rn_diff
1901    1   90  1   1   90
1901    2   90  2   1   0
1901    3   83  3   1   -7
1902    7   99  1   1   99
1902    9   87  2   2   -12
1902    8   67  3   3   -20
```

参考答案:

```
create table stu(
Stu_no int,
class string,
score int
)
row format delimited
fields terminated by '\t'
;

load data local inpath '/hivedata/stu.txt' into table stu;


select
class,
stu_no,
score,
dr,
score-nvl(lag(score) over(distribute by class sort by dr),0)
from
(select
class,
stu_no,
score,
dense_rank() over(distribute by class sort by score desc) dr
from stu) t1
where t1.dr<4
;
```


1.1.7 每个店铺的当月销售额和累计到当月的总销售额

```
数据：
店铺,月份,金额
a,01,150
a,01,200
b,01,1000
b,01,800
c,01,250
c,01,220
b,01,6000
a,02,2000
a,02,3000
b,02,1000
```

```
b,02,1500
c,02,350
c,02,280
a,03,350
a,03,250
```

编写Hive的HQL语句求出每个店铺的当月销售额和累计到当月的总销售额?

参考答案

```
create table visits(
userid int,
month string,
visits int
)
row format delimited
fields terminated by ','
;

load data local inpath '/hivedata/visits.txt' overwrite into table visits;

select
userid,
month,
visits,
max(visits) over(distribute by userid sort by month) maxvisit,
sum(visits) over(distribute by userid sort by month) totalvisit
from
(select
userid,
month,
sum(visits) visits
from visits
group by userid,month) t1
;
```


1.1.8 使用hive的hql如下

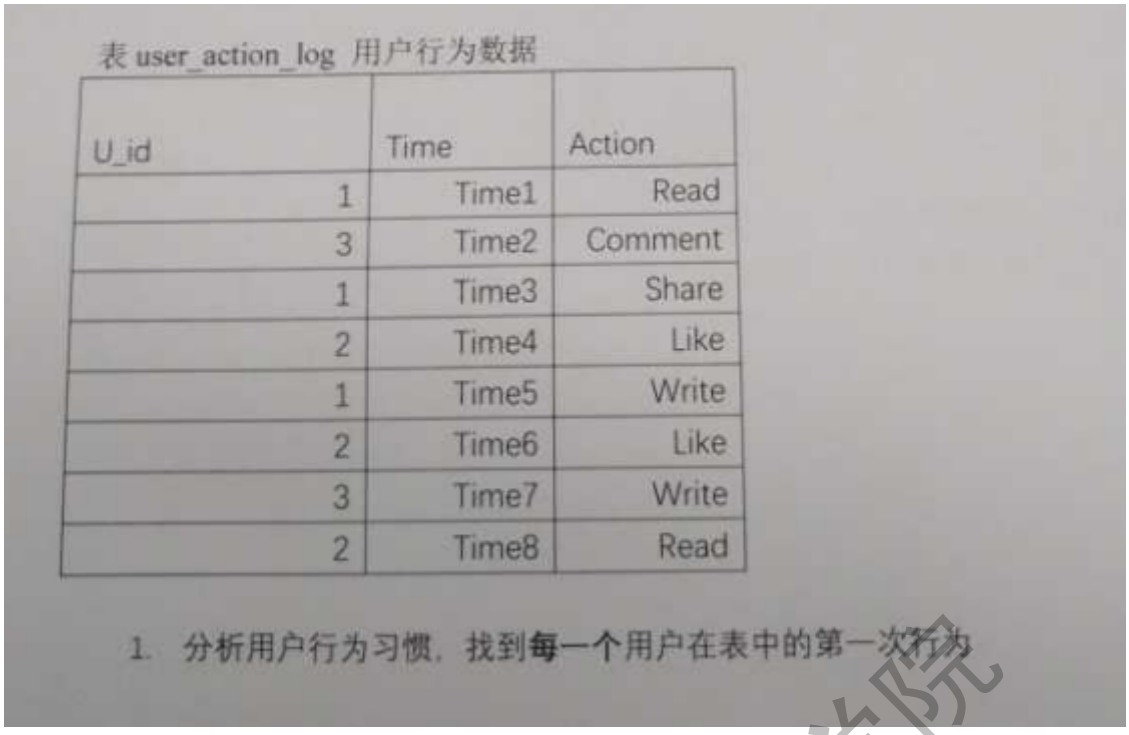


表 user_action_log 用户行为数据

U_id	Time	Action
1	Time1	Read
3	Time2	Comment
1	Time3	Share
2	Time4	Like
1	Time5	Write
2	Time6	Like
3	Time7	Write
2	Time8	Read

1. 分析用户行为习惯，找到每一个用户在表中的第一次行为

参考答案

使用代码实现

1.1.9 订单及订单类型行列互换

```
t1表:
order_id  order_type  order_time
111       N           10:00
111       A           10:05
111       B           10:10

是用hql获取结果如下:
order_id  order_type_1  order_type_2  order_time_1  order_time_2
111       N             A             10:00         10:05
111       A             B             10:05         10:10
```

参考答案

```
create table order_type(
order_id  string,
order_type  string,
order_time string
)
row format delimited
fields terminated by '\t'
;

load data local inpath '/hivedata/order_type.txt' overwrite into table
order_type;

select * from
(select
```

```
order_id,
lag(order_type) over(distribute by order_id sort by order_time) order_type_1,
order_type order_type_2,
lag(order_time) over(distribute by order_id sort by order_time) order_time_1,
order_time order_time_2
from order_type) tmp
where order_type_1 is not null
;
```

1.1.10 某APP每天访问数据存放在表access_log里面，包含日期字段ds,用户类型字段user_type，用户账号user_id,用户访问时间log_time,请使用hive的hql语句实现如下需求：

```
(1)、每天整体的访问UV、PV?
(2)、每天每个类型的访问UV、PV?
(3)、每天每个类型中最早访问时间和最晚访问时间?
(4)、每天每个类型中访问次数最高的10个用户?
```

参考答案

```
create table access_log(
ds string,
user_type string,
user_id string,
log_time string
)
row format delimited fields terminated by ','
;


(1)、每天整体的访问UV、PV?
select
count(1) as pv,
count(distinct user_id) as uv
from access_log
group by ds
;
(2)、每天每个类型的访问UV、PV?
select
user_type,
count(1) as pv,
count(distinct user_id) as uv
from access_log
group by user_type,ds
;
(3)、每天每个类型中最早访问时间和最晚访问时间?
select
first_value(log_time) over(distribute by user_type,ds sort by log_time),
last_value(log_time) over(distribute by user_type,ds sort by log_time)
from access_log
;

(4)、每天每个类型中访问次数最高的10个用户?
```

```
select * from
(select
user_type,
ds,
row_number() over(distribute by user_type,ds sort by log_time) rm
from access_log) tmp
where rm < 10
;
```

1.1.11 每个用户连续登陆的最大天数？

```
数据:
login表
uid,date
1,2019-08-01
1,2019-08-02
1,2019-08-03
2,2019-08-01
2,2019-08-02
3,2019-08-01
3,2019-08-03
4,2019-07-28
4,2019-07-29
4,2019-08-01
4,2019-08-02
4,2019-08-03

结果如下:
uid cnt_days
1 3
2 2
3 1
4 3
```

参考答案

```
create table lg_cnt(
uid int,
dt string
)
row format delimited
fields terminated by ','
;
load data local inpath '/hivedata/lg_cnt.txt' overwrite into table lg_cnt;

select
uid,
max(cn)
from
(select
uid,
count(*) cn
from
(select
```

```
uid,
date_sub(dt,row_number() over(distribute by uid sort by dt)) dt
from lg_cnt) t1
group by uid,dt) t2
group by uid
;
```

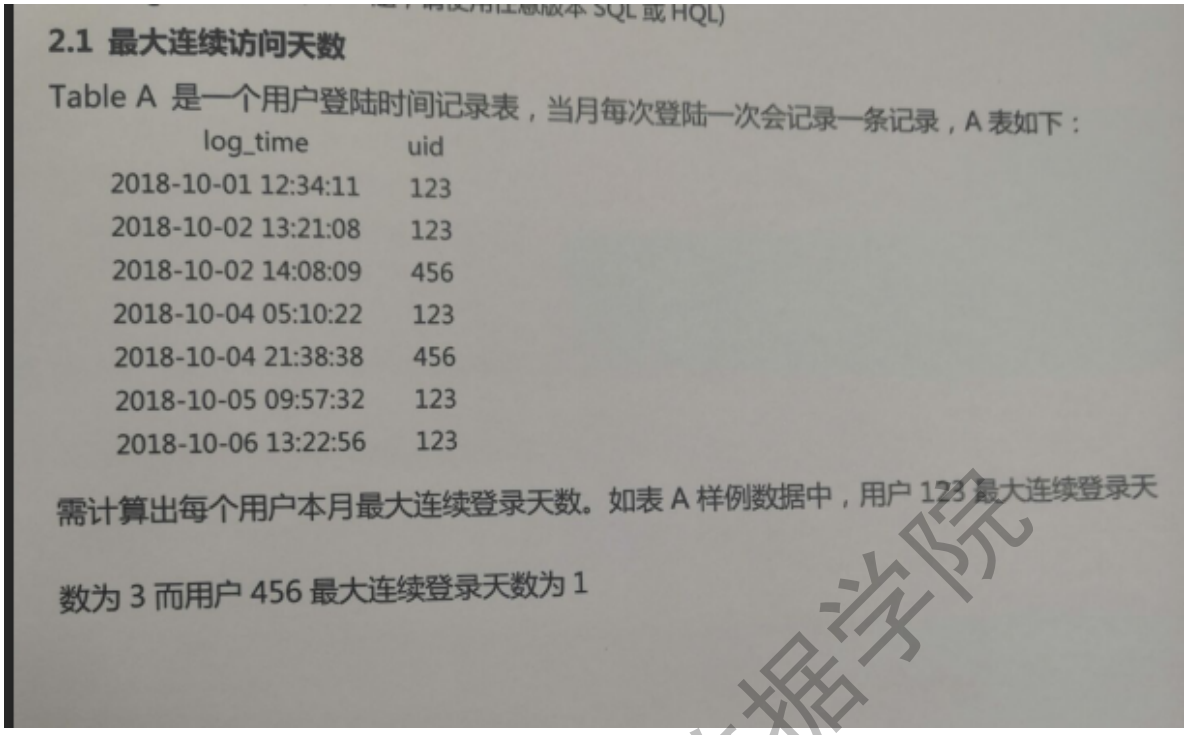
1.1.12 使用hive的hql实现男女各自第一名及其它

```
id  sex chinese_s   math_s
0   0   70  50
1   0   90  70
2   1   80  90

1、男女各自语文第一名（0:男，1:女）
2、男生成绩语文大于80，女生数学成绩大于70
```

参考答案

```
create table score_s(
id int,
sex int,
chinese_s int,
math_s int
)
row format delimited
fields terminated by '\t'
;

load data local inpath '/hivedata/score_s.txt' overwrite into table score_s;

1、select
*
from
(select
id,
sex,
chinese_s,
row_number() over(distribute by sex sort by chinese_s) rm
from score_s) tmp
where rm=1;
;


2、select
id,sex,chinese_s,math_s
from score_s
where sex=0 and chinese_s>80
union
select
id,sex,chinese_s,math_s
from score_s
where sex=1 and math_s>80
;
```

1.1.13 使用hive的hql实现最大连续访问天数



2.1 最大连续访问天数

Table A 是一个用户登陆时间记录表，当月每次登陆一次会记录一条记录，A 表如下：

log_time	uid
2018-10-01 12:34:11	123
2018-10-02 13:21:08	123
2018-10-02 14:08:09	456
2018-10-04 05:10:22	123
2018-10-04 21:38:38	456
2018-10-05 09:57:32	123
2018-10-06 13:22:56	123

需计算出每个用户本月最大连续登录天数。如表 A 样例数据中，用户 123 最大连续登录天数为 3 而用户 456 最大连续登录天数为 1

参考答案

```
log_time            uid
2018-10-01 18:00:00,123
2018-10-02 18:00:00,123
2018-10-02 19:00:00,456
2018-10-04 18:00:00,123
2018-10-04 18:00:00,456
2018-10-05 18:00:00,123
2018-10-06 18:00:00,123

create table log_times(
log_time string,
uid  string
)
row format delimited
fields terminated by ','
;

load data local inpath '/hivedata/log_times.txt' overwrite into table log_times;

select
uid,
max(cn)
from
(select
uid,
count(1) cn
from
(select
uid,
date_sub(time,row_number() over(distribute by uid sort by time)) time
```

```
from(
select
uid,
to_date(log_time) time
from
log_times
group by uid,to_date(log_time)
) t1) t2
group by uid,time) t3
group by uid
;
```

1.2 行列互换

1.2.1 对于行列互换，你有哪些解决方式，详细说明每一种方式

参考答案

具体思路需要根据数据来定，常见的解决方法如下：
行转列：
1、使用case when 查询出多列即可，即可增加列。

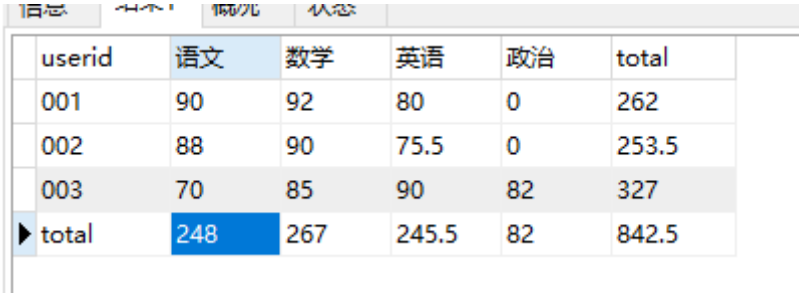
列转行：
1、lateral view explode()，使用炸裂函数可以将1列转成多行，被转换列适用于array、map等类型。lateral view posexplode(数组)，如有排序需求，则需要索引。将数组炸开成两行(索引 ， 值),需要as 两个别名。
2、case when 结合concat_ws与collect_set/collect_list实现。内层用case when，外层用collect_set/list收集，对搜集完后用concat_ws分割连接形成列。

1.2.2 编写sql实现行列互换：

数据如下：

id	userid	subject	score
1	001	语文	90
2	001	数学	92
3	001	英语	80
4	002	语文	88
5	002	数学	90
6	002	英语	75.5
7	003	语文	70
8	003	数学	85
9	003	英语	90
10	003	政治	82

编写sql实现，得到结果如下：



userid	语文	数学	英语	政治	total
001	90	92	80	0	262
002	88	90	75.5	0	253.5
003	70	85	90	82	327
total	248	267	245.5	82	842.5

参考答案

```
create table score(
id int,
sid string,
subject string,
score int
)
row format delimited
fields terminated by ','
;

load data local inpath '/hivedata/score.txt' into table score;

select
sid,
sum(case when subject="语文" then score else 0 end) as `语文`,
sum(case when subject="数学" then score else 0 end) as `数学`,
sum(case when subject="英语" then score else 0 end) as `英语`,
sum(case when subject="政治" then score else 0 end) as `政治`
from score
group by sid
;
```

1.2.3 编写sql实现如下：

```
数据：
t1表
uid tags
1   1,2,3
2   2,3
3   1,2

编写sql实现如下结果：
uid tag
1   1
1   2
1   3
2   2
2   3
3   1
3   2
```

参考答案

```
create table t1(
uid int,
tags string
)
row format delimited
fields terminated by '\t'
;
load data local inpath '/hivedata/t1.txt' into table t1;

select
uid,
tag
from t1
lateral view  explode(split(tags,",")) t2 as tag
;
```

1.2.4 用户标签连接查询

```
数据：
T1表：
Tags
1,2,3
1,2
2,3

T2表：
Id  lab
1 A
2 B
3 C

根据T1和T2表的数据，编写sql实现如下结果：
ids tags
1,2,3   A,B,C
1,2 A,B
2,3 B,C
```

参考答案

```
create table t2(
tags string
)
;
load data local inpath '/hivedata/t2.txt' overwrite into table t2;


create table t3(
id int,
lab string
)
row format delimited
fields terminated by ' '
;
load data local inpath '/hivedata/t3.txt' overwrite into table t3;
```

```
select
tags,
concat_ws(",",collect_list(lab))
from
(select
t1.tags tags,
t3.lab lab
from
(select
tags,
id
from t2
lateral view explode(split(tags,",")) tmp as id) t1
join t3
on t1.id=t3.id) tmp
group by tags
;
```

1.2.5 用户标签组合

数据：
t1表：

```
id  tag flag
a   b   2
a   b   1
a   b   3
c   d   6
c   d   8
c   d   8
```

编写sql实现如下结果：

```
id  tag flag
a   b   1|2|3
c   d   6|8
```

参考答案

```
create table t4(
id string,
tag string,
flag int
)
row format delimited
fields terminated by '\t'
;
load data local inpath '/hivedata/t4.txt' overwrite into table t4;

select
id,
tag,
concat_ws("|",collect_set(flag)) flag
from t4
```

```
group by id,tag
;
```

1.2.6 用户标签行列互换

```
数据：
t1表
uid name    tags
1   goudan  chihuo,huaci
2   mazi    sleep
3   laotie  paly

编写sql实现如下结果：
uid name    tag
1   goudan  chihuo
1   goudan  huaci
2   mazi    sleep
3   laotie  paly
```

参考答案

```
create table t5(
uid string,
name string,
tags string
)
row format delimited
fields terminated by '\t'
;
load data local inpath '/hivedata/t5.txt' overwrite into table t5;

select
uid,
name,
tag
from t5
lateral view explode(split(tags,",")) t1 as tag
;
```

1.2.7 hive实现词频统计

```
数据：
t1表：
uid contents
1   i|love|china
2   china|is|good|i|i|like

统计结果如下,如果出现次数一样，则按照content名称排序：
content cnt
```

```
i   3
china   2
good    1
like    1
love    1
is  1
```

参考答案

```
create table content(
uid int,
contents string
)
row format delimited
fields terminated by '\t'
;
load data local inpath '/hivedata/content.txt' overwrite into table content;


select
ct,
count(1) cn
from
(select
ct
from content
lateral view explode(split(contents,"\\|")) tmp as ct) tmp
group by ct
order by cn desc,ct
;
```

1.2.8 课程行转列

```
数据：
t1表
id course
1,a
1,b
1,c
1,e
2,a
2,c
2,d
2,f
3,a
3,b
3,c
3,e

根据编写sql，得到结果如下(表中的1表示选修，表中的0表示未选修)：
id a b c d e f
1 1 1 1 0 1 0
2 1 0 1 1 0 1
3 1 1 1 0 1 0
```

参考答案

```
create table course(
id int,
course string
)
row format delimited
fields terminated by ','
;
load data local inpath '/hivedata/course.txt' overwrite into table course;

select
id,
sum(case when c.course="a" then 1 else 0 end) as `a`,
sum(case when c.course="b" then 1 else 0 end) as `b`,
sum(case when c.course="c" then 1 else 0 end) as `c`,
sum(case when c.course="d" then 1 else 0 end) as `d`,
sum(case when c.course="e" then 1 else 0 end) as `e`,
sum(case when c.course="f" then 1 else 0 end) as `f`
from course c
group by id
;
```

1.2.9 兴趣行转列

```
t1表
name    sex hobby
janson  男    打乒乓球、游泳、看电影
tom 男    打乒乓球、看电影

hobby最多3个值，使用hql实现结果如下：
name    sex hobby1  hobby2  hobby3
janson  男    打乒乓球      游泳      看电影
tom 男    打乒乓球      看电影
```

参考答案

```
create table ho(
name  string,
sex  string,
hobby array<string>
)
row format delimited
fields terminated by '\t'
collection items terminated by ','
;
load data local inpath '/hivedata/hobby.txt' overwrite into table ho;

select
name,
sex,
case when true then nvl(hobby[0],"") else null end hobby1,
case when true then nvl(hobby[1],"") else null end hobby2,
case when true then nvl(hobby[2],"") else null end hobby3
```

```
from ho
;
```

1.2.10 用户商品行列互换

```
t1表:
用户    商品
A       P1
B       P1
A       P2
B       P3
请你使用hql变成如下结果:
用户     P1    P2     P3
A         1     1     0
B         1     0     1
```

参考答案

```
create table t39(
uname string,
pro string
)
row format delimited
fields terminated by '\t'
;
load data local inpath '/hivedata/t39.txt' overwrite into table t39;

select
uname,
sum(case when pro='P1' then 1 else 0 end) as P1,
sum(case when pro='P2' then 1 else 0 end) as P2,
sum(case when pro='P3' then 1 else 0 end) as P3
from t39
group by uname
;
```

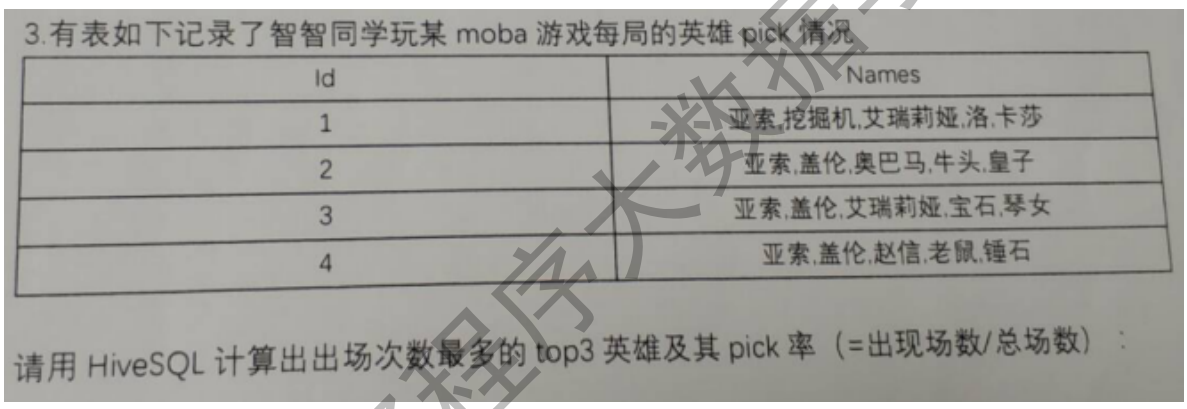
1.2.11 成绩课程行列互换

```
t1表:
name    course      score
aa      English     75
bb      math        85
aa      math        90

使用hql输出以下结果
name    English     math
aa      75          90
bb      0           85
```

参考答案

```
create table t38(
sname string,
course string,
score int
)
row format delimited
fields terminated by '\t'
;
load data local inpath '/hivedata/t38.txt' overwrite into table t38;

select
sname,
sum(case when course='English' then score else 0 end) as English,
sum(case when course='math' then score else 0 end) as math
from t38
group by sname
;
```

1.2.12 求top3英雄及其pick率



3.有表如下记录了智智同学玩某 moba 游戏每局的英雄 pick 情况

Id	Names
1	亚索,挖掘机,艾瑞莉娅,洛,卡莎
2	亚索,盖伦,奥巴马,牛头,皇子
3	亚索,盖伦,艾瑞莉娅,宝石,琴女
4	亚索,盖伦,赵信,老鼠,锤石

请用 HiveSQL 计算出出场次数最多的 top3 英雄及其 pick 率（=出现场数/总场数）：

参考答案

```
create table heros(
id int,
names array<string>
)
row format delimited
fields terminated by '\t'
collection items terminated by ','
;

load data local inpath '/hivedata/heros.txt' overwrite into table heros;

select
name,
count(1),
round(count(1)/totalcn,2)
from
(select
count(1) over() totalcn,
name
from heros lateral view explode(names) t as name) t1
group by name,totalcn
```

```
;
```

1.3 时间函数

1.3.1 常见的时间函数

参考答案

```
from_unixtime(bigint unixtime,[string format]): 时间戳转日期函数，
unix_timestamp([string date]): 转换成时间戳，然后转换格式为“yyyy-MM-dd HH:mm:ss“的日期到UNIX时间戳。如果转化失败，则返回0，返回bigint类型
to_date(string timestamp): 将时间戳转换成日期，默认格式为2011-12-08 10:03:01
year() : 将时间戳转换成年，默认格式为2011-12-08 10:03:01
month() : 将时间戳转换成月，默认格式为2011-12-08 10:03:01
hour() : 将时间戳转换成小时，默认格式为2011-12-08 10:03:01
day(string date) : 将时间戳转换成天，默认格式为2011-12-08 10:03:01
date_diff(string enddate, string startdate) : 日期比较函数，反回结束日期减去开始日期的天数
date_sub(string startdate, int days) : 日期减少函数，返回开始日期减少days天后的日期字符串
date_add(string startdate, int days) : 日期增加函数，返回开始日期增加days天后的日期字符串
last_day(string date) : 返回该月的最后一天的日期，可忽略时分秒部分(HH:mm:ss)。
last_day(string date)返回string类型的值。
next_day(string date,string x) : 返回下一个星期x的日期(x为前两英文星期前两位或者全写MONDAY)，返回字符串。
current_date() : 获取当天的日期，返回字符串，没有任何的参数。
current_timestamp() : 获取当前的时间戳
```

1.3.2 时间戳函数：unix_timestamp，from_unixtime

```
获取当前时间戳：

获取"2019-07-31 11:57:25"对应的时间戳：

获取"2019-07-31 11:57"对应的时间戳：

获取时间戳:1564545445所对应的日期和时分秒：

获取时间戳:1564545446所对应的日期和小时(yyyy/MM/dd HH):
```

参考答案

```
select unix_timestamp("2019-07-31 11:57:25","yyyy-MM-dd HH:mm:ss")

select unix_timestamp("2019-07-31 11:57","yyyy-MM-dd HH:mm")

select from_unixtime(1564545445,"yyyy-MM-dd HH:mm:ss");

select from_unixtime(1564545445,"yyyy/MM/dd HH");
```

1.3.3 时间格式转换：yyyyMMdd -> yyyy-MM-dd

```
数据:
t1表
20190730
20190731

编写sql实现如下的结果:
2019-07-30
2019-07-31
```

参考答案

```
create table dt(
dt string
);

load data local inpath '/hivedata/dt.txt' overwrite into table dt;

select
from_unixtime(unix_timestamp(dt,"yyyyMMdd"),"yyyy-MM-dd" )
from dt
;
```

1.4 交差并集

1.4.1 使用hive求出两个数据集的差集

```
数据
t1表:
id name
1 zs
2 ls

t2表:
id name
1 zs
3 ww

结果如下:
id name
```

```
2 ls
3 ww
```

参考答案

```
create table diff_t1(
id string,
name string
)
row format delimited
fields terminated by ' '
;
load data local inpath '/hivedata/diff_t1.txt' overwrite into table diff_t1;

create table diff_t2(
id string,
name string
)
row format delimited
fields terminated by ' '
;
load data local inpath '/hivedata/diff_t2.txt' overwrite into table diff_t2;

select t1.id as id,
t1.name as name
from diff_t1 t1
left join diff_t2 t2
on t1.id=t2.id
where t2.id is null
union
select
t2.id as id,
t2.name as name
from diff_t1 t1
right join diff_t2 t2
on t1.id=t2.id
where t1.id is null
;
```


1.4.2 两个表A 和B ，均有key 和value 两个字段，写一个SQL语句，将B表中的value值置成A表中相同key值对应的value值

```
A:
key vlaue
k1  123
k2  234
k3  235

B:
key value
k1  111
k2  222
k5  246
```

使用hive的hql实现，结果是B表数据如下：

```
k1  123
k2  234
k5  246
```

参考答案

```
create table t1(
key string,
value string
)
row format delimited fields terminated by " "
;
create table t2(
key string,
value string
)
row format delimited fields terminated by " "
;
load data local inpath '/data/t1' into table t1;
load data local inpath '/data/t2' into table t2;

select
a.key,
if(a.value2 is null, a.value1,a.value2) value
from
(select
t2.key,
t2.value value1,
t1.value value2
from t2
left join t1
on t1.key = t2.key
) a
;
```


1.4.3 有用户表user(uid,name)以及黑名单表Banuser(uid)

1、用left join方式写sql查出所有不在黑名单的用户信息
2、用not exists方式写sql查出所有不在黑名单的用户信息

参考答案

```
create table u(
id string,
name string
)
row format delimited
fields terminated by ','
;

create table banuser(
id string
```

```
)
;
load data local inpath '/hivedata/banuser.txt' overwrite into table banuser;
load data local inpath '/hivedata/u.txt' overwrite into table u;


1、
select
u.id,
u.name
from u
left join banuser b
on u.id=b.id
where b.id is null
;

2、
select
u.id,
u.name
from u
where not exists (select 1 from banuser b where u.id=b.id)
;
```

1.5 函数

1.5.1 hive中coalesce()、nvl()、concat_ws()、collect_list()、collect_set()、regexp_replace().这几个函数的意义

参考答案

```
coalesce(T v1, T v2, ...) : 返回列表中的第一个非空元素，如果列表元素都为空则返回NULL。
例:select coalesce(NULL,null,123,"ABC");    返回123

nvl(T v1,T v2) : 空值判断，如果v1非空则返回v1，如果v1为空，则返回v2，v1和v2需要同类型。
例：select nvl(null,1);    返回1

concat_ws(separator, str1, str2,...) : 指定分隔符(第一位)连接字符串函数。参数需要字符串。
例：select concat_ws("|","1","2","3");  返回1|2|3

collect_list(T col) :  将某列的值连接在一起，返回字符串数组，有相同的列值不会去重。通常可以使用gruop by搭配使用，但是也可以不用group by。
例：select collect_list(id) from t1;  返回将id连接在一起的字符串。如id值为1，2，2，则返回["1","2","2"]

collect_set(T col) :  将某列的值连接在一起，返回字符串数组，有相同的列值会去重。通常可以使用gruop by搭配使用，但是也可以不用group by。
例：select collect_list(id) from t1;  返回将id连接在一起的字符串。如id值为1，2，2，则返回["1","2"]
```

regexp_replace(source_string, pattern[, replace_string [, position[,occurrence, [match_parameter]]]]) : 用一个指定的 replace_string 来替换匹配的模式，从而允许复杂的"搜索并替换"操作。
例：select regexp_replace(img,".jpg","*.png") from t2; 将img列中有*.png的换成.jpg. 如img有两个数据为1.png 和 2.jsp，则返回1.jpg 和 2.jsp

1.6 聚合

1.6.1 你们公司使用什么来做的cube

参考答案

使用with cube 、 with rollup 或者grouping sets来实现cube。详细解释如下：

0、hive一般分为基本聚合和高级聚合，而基本聚合就是常见的group by，而高级聚合就是grouping set、cube、rollup等。一般group by与hive内置的聚合函数max、min、count、sum、avg等搭配使用。

1、grouping sets可以实现对同一个数据集的多重group by操作。事实上grouping sets是多个group by进行union alll操作的结合，它仅使用一个stage完成这些操作。grouping sets的子句中如果包换()数据集，则表示整体聚合。多用于指定的组合查询。

2、cube俗称是数据立方，它可以时限hive任意维度的组合查询。即使用with cube语句时，可对group by后的维度做任意组合查询，如：group a,b,c with cube ,则它首先group a,b,c 然后依次group by a,c 、 group by b,c、group by a,b 、group a 、group b、group by c、group by () 等这8种组合查询，所以一般cube个数=2^3个。2是定值，3是维度的个数。多用于无级联关系的任意组合查询。

3、rollup是卷起的意思，俗称层级聚合，相对于grouping sets能指定多少种聚合，而with rollup则表示从左往右的逐级递减聚合，如:group by a,b,c with rollup 等价于 group by a, b, c grouping sets((a, b, c), (a, b), (a), ( )).直到逐级递减为()为止,多适用于有级联关系的组合查询，如国家、省、市级联组合查询。

4、Grouping__ID在hive2.3.0版本被修复过，修复后的发型版本和之前的不一样。对于每一列，如果这列被聚合过则返回0，否则返回1。应用场景暂时很难想到用于哪儿。

5、grouping sets/cube/rollup三者的区别：
注：
grouping sets是指定具体的组合来查询。
with cube 是group by后列的所有的维度的任意组合查询。
with rollup 是group by后列的从左往右逐级递减的层级组合查询。
cube/rollup 后不能加()来选择列，hive是要求这样。

具体grouping sets、cube 、rollup的练习请参考"千峰大数据之hive练习.pdf"

1.7 正则

1.7.1 访问日志正则提取

```
表t1(注:数据时正常的访问日志数据,分隔符全是空格)
8.35.201.160 - - [16/May/2018:17:38:21 +0800] "GET
/uc_server/data/avatar/000/01/54/22_avatar_middle.jpg HTTP/1.1" 200 5396

使用hive的hql实现结果如下:
ip  dt  url
8.35.201.160    2018-5-16 17:38:21
/uc_server/data/avatar/000/01/54/22_avatar_middle.jpg
```

参考答案

```
create table login_log(
log  string
)

load data local inpath '/hivedata/login_log.txt' overwrite into table login_log;

select
regexp_extract(log,"([0-9.]+\\d+) - - \\[(.+ \\+\\d+)\\] .+(GET) (.+) (HTTP)\\S+
(\\d+) (\\d+)",1),
from_unixtime(unix_timestamp(regexp_extract(log,"([0-9.]+\\d+) - - \\[(.+
\\+\\d+)\\] .+(GET) (.+) (HTTP)\\S+ (\\d+) (\\d+)",2),"dd/MMM/yyyy:HH:mm:ss
Z"),"yyyy-MM-dd HH:mm:ss"),
regexp_extract(log,"([0-9.]+\\d+) - - \\[(.+ \\+\\d+)\\] .+(GET) (.+) (HTTP)\\S+
(\\d+) (\\d+)",4)
from login_log
where log regexp "([0-9.]+\\d+) - - \\[(.+ \\+\\d+)\\] .+(GET) (.+) (HTTP)\\S+
(\\d+) (\\d+)"
;
```

1.8 优化

1.8.1 你做过hive的那些优化

参考答案

```
hive的优化是一个老生常谈的话题。具体优化的点可以考虑一下几方面:
1、硬件优化(运维部加资源，一般难实现，除非集群资源(可视化界面或者命令查看)真的不够)
2、环境优化(包括hive及相关使用引擎属性优化，具体常用属性如下)
3、业务优化(待更新)
4、语句优化(待更新)

相关属性设置如下:
cube炸裂优化:
hive.map.agg=true;      #最好将其设置为true，因为会用到ma端聚合。
hive.new.job.grouping.set.cardinality=30;    #在grouping sets/cube/rollup中是否启用
新的job来执行，主要是因为如果分组多而造成数据立方体炸裂，适当设置该阀值，默认为30.

本地模式:
hive.exec.mode.local.auto=true; 默认false
hive.exec.mode.local.auto.inputbytes.max=134217728;
hive.exec.mode.local.auto.input.files.max=4;
```

合并文件:
hive.merge.mapfiles=true; map端合并小文件
hive.merge.mapredfiles=false; 是否在mapreduce输出进行小文件合并
hive.merge.size.per.task=256000000; mapreduce的的输出每个任务合并文档的大小
hive.merge.smallfiles.avgsize=16000000; 如果输出文件数量平均值小于该值，则默认开启输出文件合并

jvm重用:
mapreduce.job.jvm.numtasks=1; 1个jvm里面运行多少个task，如-1则表示没有限制

map端聚合(map aggr):
hive.map.aggr=true; group by语句是否进行map端聚合，类似combiner，重复key不多会影响效率

queue资源队列:
mapreduce.job.queuename=default; job被提交的queue队列，默认是default，队列参考mapred-queues.xml,注意其队列是否存在和当前用户是否有权限提交job到该队列。

parallel并行:
hive.exec.parallel=false; 是否允许多个job并行执行
hive.exec.parallel.thread.number=8; 最多允许8个job并行执行

map-join:
hive.auto.convert.join=true; 是否允许自动将普通join转换为map端join，以此达到优化
hive.mapjoin.smalltable.filesize=25000000; 小表大小
hive.auto.convert.join.noconditionaltask=true; 是否基于小表大小，同时n-1个表或者分区小于指定的表，则直接不生成额外任务。
hive.auto.convert.join.noconditionaltask.size=10000000; 转成mapjoin不生成额外作业的大小阀值。

bucket join:
hive.auto.convert.join=true;
hive.optimize.bucketmapjoin=false; 是否尝试分桶map端join来优化

sort merge bucket join:
hive.input.format=org.apache.hadoop.hive.ql.io.CombineHiveInputFormat; 输入格式,可设置为..BucketizedHiveInputFormat
hive.auto.convert.sortmerge.join=false;
如果被join的表通过略sort-merge join的标准，是否自动转换join为sortmerge的join。
(sortmerge join：)
hive.optimize.bucketmapjoin=false; 是否尝试分桶map端join来优化
hive.optimize.bucketmapjoin.sortedmerge=false; 是否尝试对分桶map端join进行合并排序
hive.auto.convert.sortmerge.join.noconditionaltask= 该属性hive1.1.0中暂时未有
hive.auto.convert.sortmerge.join.to.mapjoin=false; 是否将sortmerge join(需要开启hive.auto.convert.sortmerge.join)转换成mapjoin。

sort merge bucket map join:
需要sort merge bucket join:的全部属性
hive.mapjoin.bucket.cache.size=100; map端join的桶缓存大小
hive.auto.convert.sortmerge.join.bigtable.selection.policy=org.apache.hadoop.hive.ql.optimizer.AvgPartitionSizeBasedBigTableSelectorForAutoSMJ;
该策略将会自动将大表转换为mergesort join,默认情况下，表的最大分区将会被处理。所有策略如下:
. based on position of the table - the leftmost table is selected
org.apache.hadoop.hive.ql.optimizer.LeftmostBigTableSMJ.
. based on total size (all the partitions selected in the query) of the table
org.apache.hadoop.hive.ql.optimizer.TableSizeBasedBigTableSelectorForAutoSMJ.

```
. based on average size (all the partitions selected in the query) of the table
org.apache.hadoop.hive.ql.optimizer.AvgPartitionSizeBasedBigTableSelectorForAuto
SMJ.


skew join:
hive.optimize.skewjoin=false;  是否允许倾斜join进行优化，它的优化是：在运行时，决定造成倾斜的最大的key，然后将其分离开来单独处理，然后处理不被倾斜的key，而倾斜的key单独提出来处理会更快，因为他们可能会被转出map join。
hive.skewjoin.key=100000;  这是咱们常说的，key不均匀造成倾斜，相同key大于100000个则可认为是倾斜的key。
hive.groupby.skewindata=false;  group by查询语句中，倾斜数据是否需要优化。前提得是有有group by语句且group by的key是倾斜状态。如果该值为true，hive不支持多列上的去重操作，会报错。

hive input fromat:
hive.inpupt.format=org.apache.hadoop.hive.ql.io.CombineHiveInputFormat;  输入格式,可设置为

Analyze优化：
Analyze分析表:（也称为计算统计信息）是一种内置的Hive操作，可以执行该操作来收集表上的元数据信息。这可以极大的改善表上的查询时间，因为它收集构成表中数据的行计数，文件计数和文件大小（字节），并在执行之前将其提供给查询计划程序。
```

1.9 普通查询

1.9.1 使用hive的hql查询用户所在部门

```
dpt表
dpt_id dpt_name
1   产品
2   技术

user_dpt表
user_id dpt_id
1   1
2   1
3   2
4   2
5   3

result表
user_id dpt_id dpt_name
1       1       产品
2       1       产品
3       2       技术
4       2       技术
5       3       其他部门
```

参考答案

```
create table dpt(
dpt_id int,
dpt_name string
```

```
)
row format delimited
fields terminated by '\t'
;
load data local inpath '/hivedata/dpt.txt' overwrite into table dpt;

create table user_dpt(
user_id int,
dpt_id int
)
row format delimited
fields terminated by '\t'
;
load data local inpath '/hivedata/user_dpt.txt' overwrite into table user_dpt;


select
t1.user_id,
t1.dpt_id,
nvl(t2.dpt_name,"其他部门")
from user_dpt t1
left join dpt t2
on t1.dpt_id=t2.dpt_id
;
```

1.9.2 查出每个学期每门课程最高分记录

```
course_score表:
id,userid,course,score,term
1,zhangsan,数学,80,2015
2,lisi,语文,90,2016
3,lisi,数学,70,2016
4,wangwu,化学,80,2017
5,zhangsan,语文,85,2015
6,zhangsan,化学,80,2015

编写sql完成如下查询，一次查询实现最好，也可以写多次查询实现：
1、查出每个学期每门课程最高分记录(包含全部5个字段)
2、查出单个学期中语文课在90分以上的学生的数学成绩记录(包含全部5个字段)
```

参考答案

```
create table course_score(
id string,
name string,
course string,
score int,
year string
)
row format delimited
fields terminated by ','
;

load data local inpath '/hivedata/course_score.txt' overwrite into table
course_score;
```

```
1、
select
s.id,
s.name,
s.course,
s.score,
s.year
from
course_score s
join
(select
course,
year,
max(score) score
from course_score
group by course,year) t1
on s.course=t1.course
and
s.year=t1.year
and
s.score=t1.score
;

2、
select
s.id,
s.name,
s.course,
s.score,
s.year
from
course_score s
join
(select
id,
name,
course,
score,
year
from
course_score
where
score >=70
and
course="语文") t1
on s.name=t1.name
where s.course="数学"
;
```


1.9.3 设计数据库表，用来存放学生基本信息，课程信息，学生的课程及成绩，并给出sql语句，查询平均成绩大于85的所有学生

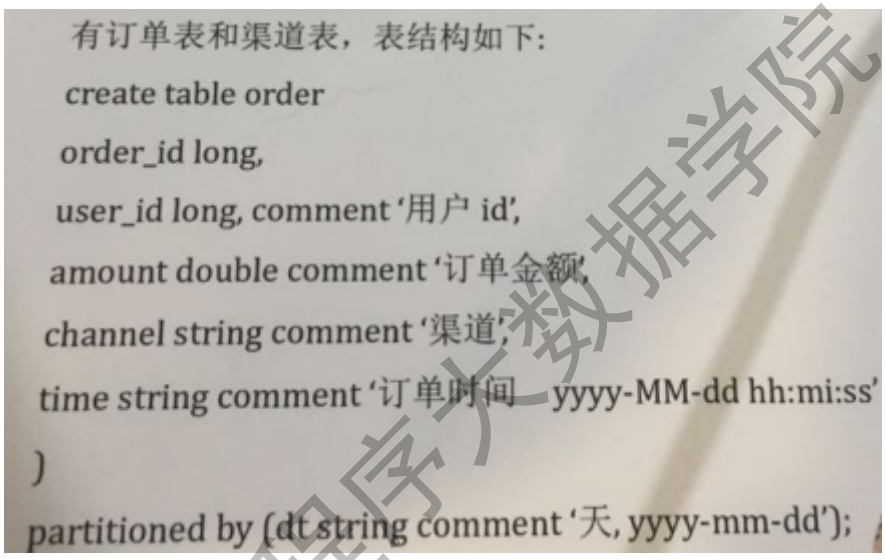
参考答案

```
stu_1
id,name,age,addr
1,zs1,22,bj
2,zs2,22,bj
3,zs3,22,bj
4,zs4,22,bj
5,zs5,22,bj

course_1
cid,cname
1,语文
2,数学
3,政治
4,美术
5,历史

course_sc
id,cid,score
1,1,87
1,2,92
1,3,69
2,2,83
2,3,92
2,4,87
2,5,83


create table stu_1(
id string,
name string,
age int,
addr string
)
row format delimited
fields terminated by ','
;


create table course_1(
cid string,
cname string
)
row format delimited
fields terminated by ','
;

create table course_sc(
id string,
cid string,
score int
)
row format delimited
fields terminated by ','
;

load data local inpath '/hivedata/course_1.txt' overwrite into table course_1;
load data local inpath '/hivedata/stu_1.txt' overwrite into table stu_1;
load data local inpath '/hivedata/course_sc.txt' overwrite into table course_sc;
```

```
select
cs.id,
avg(score) avgscore
from course_sc cs
group by cs.id
having avgscore>85
;
```

1.9.4 每个渠道的下单用户数、订单总金额

有一个订单表和渠道表，结构如下：



请使用hive hql查询出2019-08-06号 每个渠道的下单用户数、订单总金额。
hql语句实现，结果表头如下：
channel user_num order_amount

参考答案

```
create table order_34(
order_id bigint,
user_id int,
amount double,
channel string,
time string comment 'yyyy-MM-dd HH:mm:ss'
)
partitioned by(dt string comment 'yyyy-MM-dd')
row format delimited
fields terminated by ','
;


1,100,19,a,2019-08-06 19:00:00
2,101,19,b,2019-08-06 19:00:01
3,100,19,a,2019-08-05 19:00:00
4,101,19,b,2019-08-05 19:00:01
```

```
5,102,19,a,2019-08-06 19:00:00
6,102,19,a,2019-08-06 19:00:01

load data local inpath '/hivedata/order_34.txt' into table order_34
partition(dt='2019-08-06');

hive hql查询出2019-08-06号 每个渠道的下单用户数、订单总金额。
select
channel,
count(distinct user_id),
sum(amount)
from order_34
where dt='2019-08-06' and to_date(time)='2019-08-06'
group by channel,to_date(time)
;
```

1.9.5 登录且阅读的用户数，已经阅读书籍数量及其它

```
有如下三张表:
表A(登录表):
ds  user_id
2019-08-06  1
2019-08-06  2
2019-08-06  3
2019-08-06  4

表B(阅读表):
ds  user_id read_num
2019-08-06  1   2
2019-08-06  2   3
2019-08-06  3   6

表C(付费表):
ds  user_id price
2019-08-06  1   55.6
2019-08-06  2   55.8

基于上述三张表，请使用hive的hql语句实现如下需求:
(1)、用户登录并且当天有个阅读的用户数，已经阅读书籍数量
(2)、用户登录并且阅读，但是没有付费的用户数
(3)、用户登录并且付费，付费用户书籍和金额
```

参考答案

```
create table if not exists lg(
ds string,
user_id string
)
row format delimited
fields terminated by ' '
;

load data local inpath '/hivedata/lg.txt' into table lg;

create table if not exists read(
ds string,
```

```
user_id string,
read_num  int
)
row format delimited
fields terminated by ' '
;

load data local inpath '/hivedata/read.txt' into table read;

create table if not exists pay(
ds string,
user_id string,
price  double
)
row format delimited
fields terminated by ' '
;

load data local inpath '/hivedata/pay.txt' into table pay;

(1)、用户登录并且当天有阅读的用户数，已经阅读书籍数量
select
count(1)
from read r
join lg l
on l.user_id = r.user_id and l.ds = r.ds
group by r.ds;

(2)、用户登录并且阅读，但是没有付费的用户数

select
r.ds ds,
count(1)
from read r
left semi join lg l
on l.user_id = r.user_id and l.ds = r.ds
left join pay  p
on r.ds=p.ds and r.user_id=p.user_id
where p.user_id is null
group by r.ds
;


(3)、用户登录并且付费，付费用户id和金额
select
p.user_id,
p.price
from pay p
left join lg
on p.user_id=lg.user_id and p.ds=lg.ds
where lg.user_id is not null
;
```


1.9.6 高消费者报表

有三个表，分别是：
区域(district) 区域中有两个字段分别是区域Id(disid)和区域名称(disname)
城市(city) 城市有两个字段分别是城市ID(cityid)和区域ID(disid)
订单(order) 订单有四个字段分别是订单ID(orderid)、用户ID(userid)、城市ID(cityid)和消费金额(amount)。

district表：
disid disname
1 华中
2 西南

city表：
cityid disid
1 1
2 1
3 2
4 2
5 2

order表：
oid userid cityid amount
1 1 1 1223.9
2 1 1 9999.9
3 2 2 2322
4 2 2 8909
5 2 3 6789
6 2 3 798
7 3 4 56786
8 4 5 78890

高消费者是消费金额大于1W的用户，使用hive hql生成如下报表：
区域名 高消费者人数 消费总额

参考答案

```
create table district(
disid string,
disname string
)
row format delimited fields terminated by '\t'
;

create table city(
cityid string,
disid string
)
row format delimited fields terminated by '\t'
;

create table order_29(
oid string,
userid string,
cityid string,
amount double
)
row format delimited fields terminated by '\t'
;
```

```
load data local inpath '/hivedata/district.txt' into table district;
load data local inpath '/hivedata/city.txt' into table city;
load data local inpath '/hivedata/order_29.txt' into table order_29;


select
b.disid,
c.disname,
a.userid,
sum(a.amount)
from order_29 a
join city b on a.cityid=b.cityid
join district c on b.disid = c.disid
group by b.disid,c.disname,a.userid
;
```

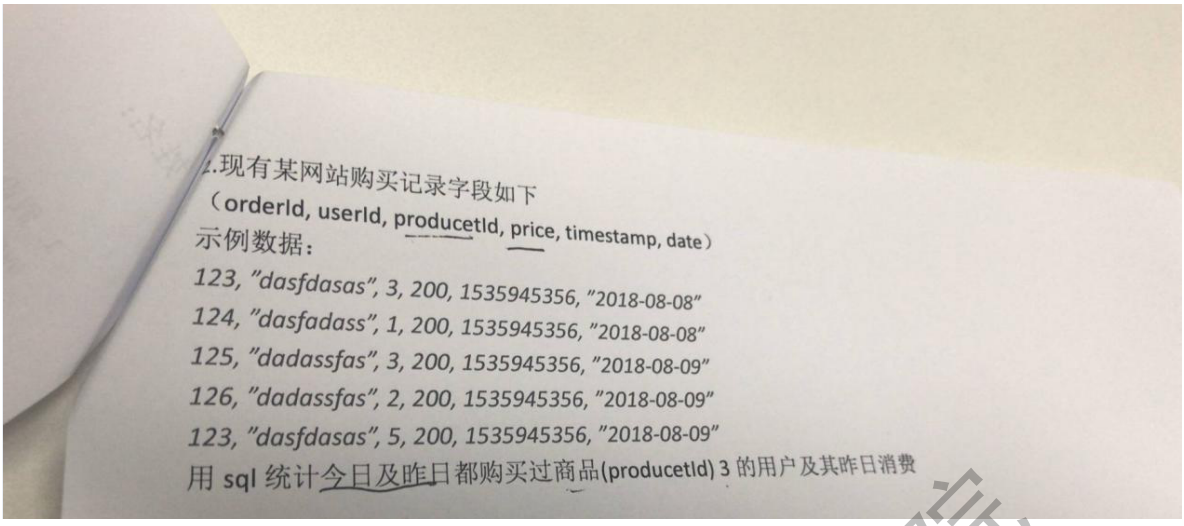
1.9.7 请使用sql计算pv、uv

```
数据:
t1表
uid date    url
1   2019-08-06  http://www.baidu.com
2   2019-08-06  http://www.baidu.com
3   2019-08-06  http://www.baidu.com
3   2019-08-06  http://www.soho.com
3   2019-08-06  http://www.meituan.com
3   2019-08-06

结果如下:
date uv pv
2019-08-6 3 5
```

参考答案

```
create table uv_pv(
uid string,
dt string,
url string
)
row format delimited
fields terminated by ','
;

load data local inpath '/hivedata/uv_pv.txt' overwrite into table uv_pv;

select
dt,
count(distinct uid),
count(url)
from uv_pv
group by dt
```

1.9.8 使用hive的hql实现买过商品3的用户及其昨日消费：



参考答案

```
orderid userid  productid   price   timestamp   date
123,00101,3,200,1535945356,2019-08-28
124,00100,1,200,1535945356,2019-08-28
125,00101,3,200,1535945356,2019-08-29
126,00101,2,200,1535945356,2019-08-29
127,00102,5,200,1535945356,2019-08-29
128,00103,3,200,1535945356,2019-08-29
129,00103,3,200,1535945356,2019-08-29

create table buy_log(
orderid int,
userid string,
productid int,
price double,
tm string,
dt string
)
row format delimited
fields terminated by ','
;

load data local inpath '/hivedata/buy_log.txt' overwrite into table buy_log;



select
userid userid,
sum(case when dt=date_sub(current_date,1) then price else 0 end) pay
from
(select
userid,
price,
dt
from buy_log
where productid=3 and (dt=current_date or dt=date_sub(current_date,1))) t1
group by userid,dt
```

```
having pay>0
;
```

1.9.9 统计为用户配重了角色的用户角色数量

```
用户表和角色表，统计为用户配重了角色的用户角色数量（hql）
用户表t1:
uid uname   role
1   zs  1,1,2,2,2
2   ls  1,2
3   ww  2,3,3

角色表t2:
roleid  rolename
1,唤醒师
2,召唤师
3,魔法师

结果如下:
uid uname   roleid  cnt
1   zs  1   召唤师   2
1   zs  2   唤醒师   3
3   ww  3   魔法师   2
```

参考答案

```
create table user_role(
uid int,
uname string,
role array<int>
)
row format delimited
fields terminated by '\t'
collection items terminated by ','
;

load data local inpath '/hivedata/user_role.txt' overwrite into table user_role;

create table role(
roleid  int,
rolename string
)
row format delimited
fields terminated by ','
;

load data local inpath '/hivedata/role.txt' overwrite into table role;


select
tmp.uid,
tmp.uname,
tmp.roleid,
role.rolename,
count(1)
from role join
```

```
(select
uid,
uname,
roleid
from user_role
lateral view explode(role) t as roleid) tmp
on tmp.roleid=role.roleid
group by tmp.uid,tmp.uname,tmp.roleid,role.rolename
;
```

1.10 其它

1.10.1 Hive是否发生过数据倾斜，怎么处理的，原理是什么

参考答案

发生过数据倾斜。

怎么处理？
先预判断是什么问题造成的倾斜，常见的有group by 、count(distinct) 、 join等情形产生。
1、如果是group by产生的，则可考虑设置如下属性：
set hive.map.aggr=true
set hive.groupby.skewindata=true
原理：
hive.map.aggr=true 这个配置项代表是否在map端进行聚合，类似于combiner做提前聚合。
hive.groupby.skewindata=true 这个配置为true，代表生成的查询计划会有两个 MR Job。第一个 MR Job 中，Map 的输出结果集合会随机分布到 Reduce 中，每个 Reduce 做部分聚合操作，并输出结果，这样处理的结果是相同的 Group By Key 有可能被分发到不同的 Reduce 中，从而达到负载均衡的目的；第二个 MR Job 再根据预处理的数据结果按照 Group By Key 分布到 Reduce 中（这个过程可以保证相同的 Group By Key 被分布到同一个 Reduce 中），最后完成最终的聚合操作。

2、count(distinct)产生的
如果数据量非常大，执行如select a,count(distinct b) from t group by a;类型的SQL时，会出现数据倾斜的问题。
原理：使用sum...group by代替。如select a,sum(1) from (select a, b from t group by a,b) group by a;

3、join 产生的
找出产生倾斜的key(单个key达到100000)，然后对倾斜的key进行处理
法一、 将倾斜的key单独提出来，然后进行单独处理，然后在用union all连接处理
法二、 给空值分配随机的key值，保证业务不会受影响，然后在进行join

1.10.2 Hive中什么时候使用过array和map，为什么使用

参考答案

1、源数据中某字段的值呈数组类型的，即该字段可使用array类型，比如a字段值为1,3,6.则a字段可使用array类型。
2、源数据中某字段的值呈key-value类型的，即可使用map类型，比如a字段的值为k1=v1,k2=v2.则a字段可以使用map类型。
3、josn格式中的数据也可以使用到。因为json中经常存在数组和key-value类型的数据。

1、方便取数。比如可以固定取某个下标或者某个key的值，可以取出所有key或者value值，可以进行排序等操作。

1.10.3 hive的hql中，left outer join和left semi join的区别

参考答案

共同点:
1、left outer join 和 left semi join都以左表为准，连接右表。
不同点:
1、left outer join连接不上的右表信息用NULL替代，而left semi join右表连接不上，则左表信息不会出来。
2、left outer join可以查询左右表的所有信息，而left semi join只能查询出左表信息。
3、left semi join比left outer join的效率要高。
4、场景应用上有不同。由于hive不支持exists in，semi join可以用于解决exist in的场景，比outer join高效。outer join用于其它查询场景。

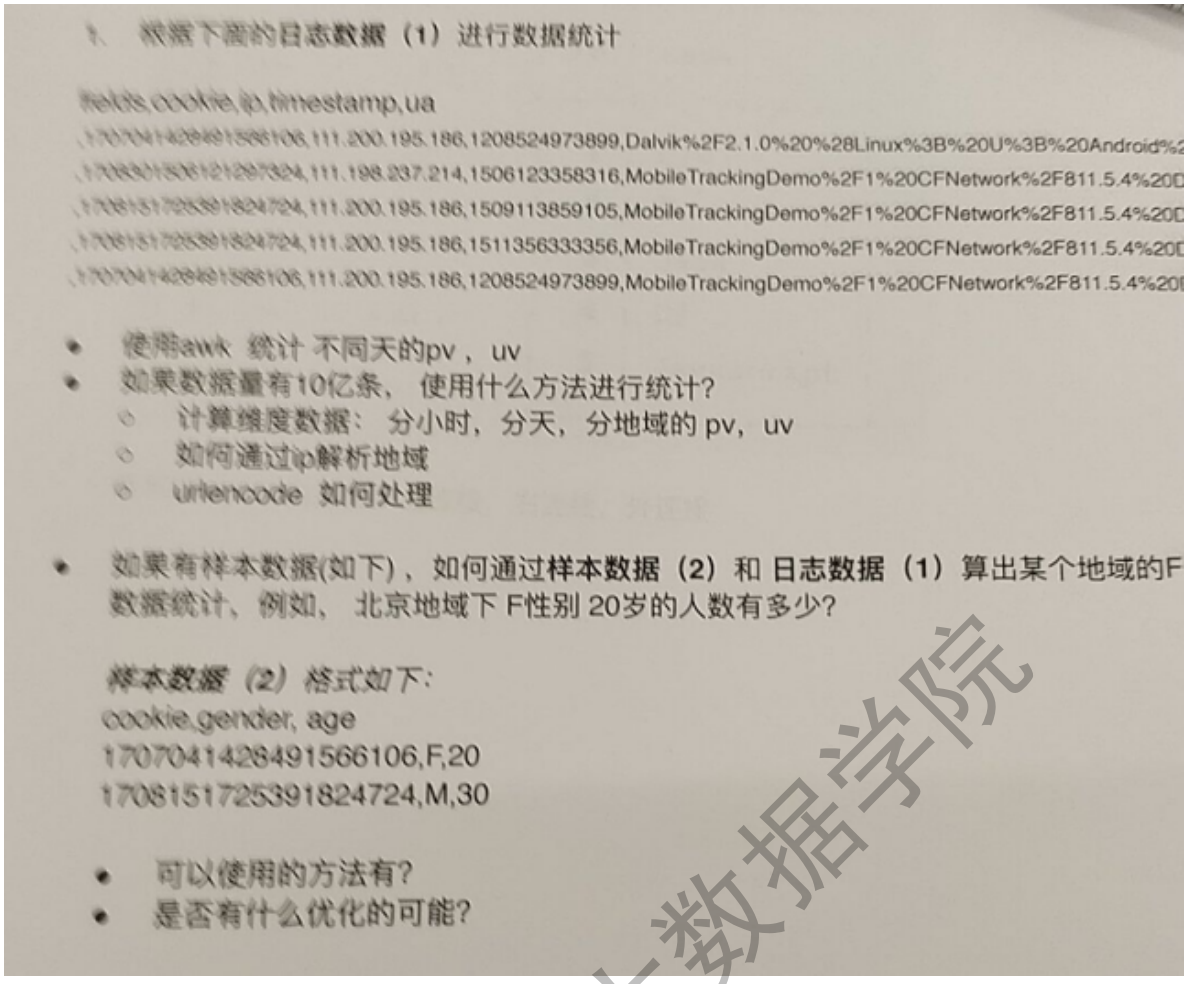
1.10.4 一张大表A(上亿条记录)和小表B(几千条记录)，如果join出现数据倾斜，有什么解决办法

参考答案

1、考虑业务上优化(就是避免这样的连接大小表join查询)，比如先把A表的无效数据过滤、对B表分批次进行join，然后将结果union all等。
2、先统计A表中的关联字段个数，看其统计值大于100000以上的有多少个值，这样的值会被认为是倾斜key，也可以适当改大其数值，让hive不认为是倾斜数据和join。
3、对第2步中的大于十万的key值所对应于B中的值抽取出来，然后和A表进行关联，然后将其各部分值进行union all。
4、考虑配置属性倾斜数据和倾斜join的相关属性。

1.10.5 统计不同天的pv、uv及其它

样例数据:
t1表
gender,cookie,ip,timestampe,ua
F,1707041428491566106,111.200.195.186,1208524973899,Dalvik%2F2.1.0%20%28Linux%3B%20U%3B%20Android
...具体数据如下图



将图片中的awk修改为使用sql编写，然后将上诉题作出回答？

参考答案

```
create table log_uv_pv(
gender string,
cookie string,
ip string,
tm string,
ua string
)
row format delimited
fields terminated by ','
;


create table log_uv_pv_1(
cookie string,
gender string,
age int
)
row format delimited
fields terminated by ','
;
```

使用自定义函数parse()将ip解析成地域信息，将时间戳转换成时间，格式为 "yyyy-MM-dd HH"，
然后按照时间和地域分组，count(ip)为UV数量，count(url)为pv数量

可以使用纯真数据库解析IP,淘宝解析IP或者自己收集IP进行解析

```
uriencode 要进行转码

select
parse(ip) region,
a.gender,
b.age,
count()
from log_uv_pv a join log_uv_pv1 b
on a.cookie=b.cookie
group by a.gender,b.age,parse(ip) region
```