Advanced SQL实战应用

如何用SQL计算留存

作者有话说：学完SQL那门课的时候，我以为自己可以算“熟练运用”了，但是经历过项目、秋招后才发现写好SQL并不是那么容易的一件事。缺乏互联网行业的基础知识让我在无数笔试中忘SQL兴叹，最终败下阵来。在这个系列里，我决定总结记录一下SQL在计算一些指标时的高级用法。

**什么是用户留存？**

互联网行业认可的留存用户的认定标准：用户不论从哪种渠道开始进入并使用产品，在一段时间周期后，该用户依然继续还在使用该产品，则该用户被认做为“留存用户”。

举个栗子，假设杨猫猫周一观看了本公众号的文章并关注，并在周二再次观看了文章，则她是保留用户。反之，如果她第一次看完文章后没有关注也没有在周二接着看其他文章，则她是一名失效用户。用户流失很糟糕。如果你是一个创业公司，你就知道”Retention is King”。作为数据分析师应该始终监控用户留存率，并探索影响留存率波动的原因。

**留存率的计算**

留存率定义为用户在某段时间内开始使用网站/应用（一般定义是注册），经过一段时间后，仍然继续使用的人被认作是留存用户。留存率体现了网站/应用的质量和保留用户的能力。

留存率计算方法：N日留存率=N日留存用户数/新增用户数\*100%

（一般统计周期为天，常见的周期维度有次日、7日、30日）

比如：

次日留存率：（第一天新增用户数，第2天还登录的用户数）/第一天总注册用户数

7日留存率：（第一天新增用户数，第8天还登录的用户数）/第一天总注册用户数

30日留存率：（第一天新增用户数，第31天还登录的用户数）/第一天总注册用户数

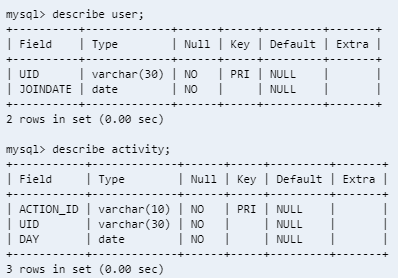
**留存率在SQL中是如何实现的？**

首先，我们来看下数据库中的内容，示例数据在文末可下载。

示例数据是我从网上找来的某网站2009年至2010年的用户登录数据（activity.csv）和用户的注册信息（user.csv）（为了方便计算，我做了部分修改）。

User表中列UID代表用户的唯一标志（账户），列JOINDATE代表用户的注册日期。

Activity表中列ACTION\_ID代表每个用户每一次登录的标志，列DAY代表用户登录的日期。



Step 1: 计算在t1时刻处于活动状态的用户，然后计算在t2时刻留存的用户数。在SQL中可以用self join方法，以次日留存为例：

select \*

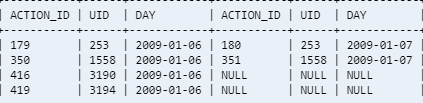
from activity

left join activity as future\_info on

activity.UID = future\_info.UID

and activity.DAY= subdate(future\_info.DAY,1);

Join之后的表每一行都包含了t1和t2时刻的登录情况，t2没有留存则为null。以2009-01-06为例：



Step 2: 根据以上join之后的表格计算去重后的活跃用户数、留存用户数和留存率，以次日留存为例：

select

activity.DAY,

count(distinct activity.UID) as active\_user, # 某日活跃用户，去重

count(distinct future\_info.UID) as retained\_user, # 次日留存用户，去重

round(count(distinct future\_info.UID)/count(distinct activity.UID),2) as retention\_rate #留存率

from activity

left join activity as future\_info

on activity.UID= future\_info.UID

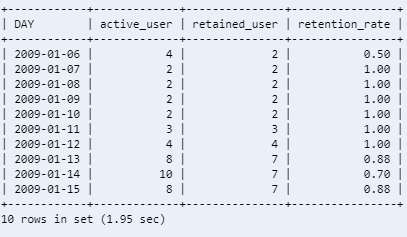
and activity.DAY=subdate(future\_info.DAY,1) # 指定留存周期，这里是次日留存所以取相差一天的活动

group by activity.DAY

order by activity.DAY

limit 10;

运行结果如下：



可以看出该网站2009年1月份的留存率非常高，在50%到100%之间波动，这是一个好现象。我们可以继续计算7日和30日留存率追踪该网站长期的用户留存，判断是否符合Facebook平台流传出的留存率“40-20-10”规则。这里就不再提供代码。

**新、老用户留存率计算**

刚刚注册的新用户与忠诚的长期用户相比，两者的留存率是完全不同的。所以我们可以分别计算各自的留存率分开比较。要计算新用户的留存，只需要在上述代码中加入条件限制活动的日期，以次日留存为例：

select

activity.DAY,

count(distinct activity.UID) as active\_user,

count(distinct future\_info.UID) as retained\_user,

count(distinct future\_info.UID)/count(distinct activity.UID) as retention\_rate

from

activity

join user

on activity.UID=user.UID

and

activity.DAY=user.JOINDATE

left join activity as future\_info

on activity.UID=future\_info.UID

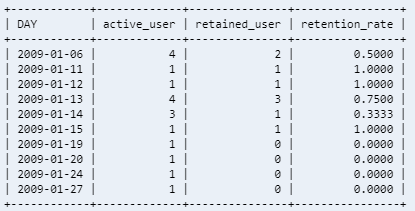
and

activity.DAY=subdate(future\_info.DAY,1)

group by activity.DAY

order by activity.DAY

limit 10;



这里我们可以看出13号之后的新用户次日留存率逐渐降低，19号之后留存率降为零。这是很有可能发生的情况，比如某公司在11日做了很多拉新、推广的活动，带来了很多新用户，但是最终留下的用户不一定在增长，甚至有可能在减少，用户如果无法从网站获得自己想要的资源就会逐渐流失。想要获得用户的认同感、有期待、有惊喜，就必须创造内容提高内容质量。

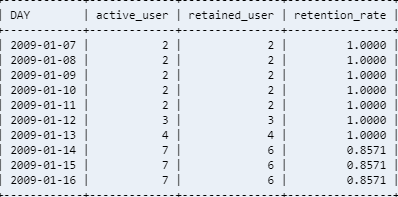
要查看老用户留存，只需更改：

User.JOINDATE=activity.DAY

至：

User.JOINDATE!=activity.DAY

这有效地排除了当天加入的用户的活动。



如图所示，老用户的留存率高于整体平均水平。

**Cohort Analysis in SQL**

Cohort Analysis是将某一个时期内的用户划分为一个cohort，并将多个cohort进行时间上的某个属性的比较的一种分析方法。Cohort Analysis在有些场景下非常有用。比如一个网站或App，在某个连续的4周里陆续更新或新增了一个功能或设计，想要知道这些功能和设计上的改动对用户的影响，就可以将每周的新注册作为一个cohort，观察这4个cohort在接下来的一段实际里的行为数据，就可以很清楚地观察到4个改动的影响。

分组留存率计算起来比较简单，对用户进行分组，先维度分，再粒度分。

什么是维度？如果按用户的首次购买时间对用户进行分组，购买时间就是维度；所谓粒度，就是时间维度是按日、周、月、季度，还是按年为分析的颗粒度？也即粒度上的差异。维度也可以是新增用户的渠道，那么新增的来源产品，还是来源的具体网址，这也属于粒度上的差异。通过这两方面的分组可以将对比的差异值逐级锁定，寻找数据表象差异背后的原因。

通常来讲，对于用户的留存分析有两个比较常用的的分组方向：

1. 从用户的获取角度分组
2. 从用户的行为角度分组

从用户获取的角度分组

如果按用户购买的时间分，粒度可选日、周、月、季度等，我们就可以分析后看出，各个时间段获取的客户在留存率表现上是不是有差异，例如在一个时间段内留存曲线是收敛并趋于稳定，还是强烈。通过留存率分析我们能够对流失的用户做用户画像分析，并在流失率高的时间段进行适当的干预。

在获取用户的角度，还可以对获客渠道进行划分，分析不同渠道的用户后续的留存情况。通过渠道维度的分析也可以判定不同渠道表现的优劣情况 ，好的渠道可以加大促销或者广告投入，差的渠道可以淘汰不用。

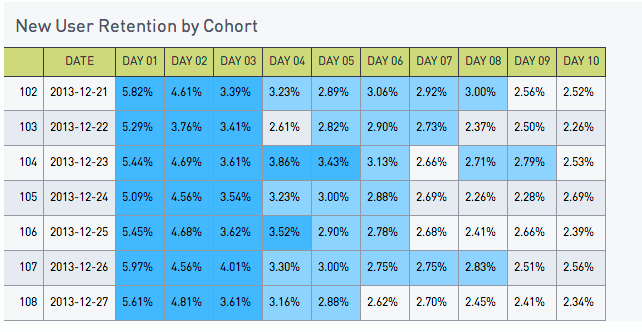
从用户行为角度分组

从用户行为角度分组对于功能比较复杂的产品也非常重要。比如一个浏览器产品，已不单单是解决用户访问网页的需求，通过浏览器，还可以向用户推送新闻、小说、视频等服务，浏览器俨然已经变成了一个内容的综合载体。一个用户选择用一个浏览器来看小说，并不代表他也一定会用浏览器来看新闻或视频。所以同样是用户流失，可能是对产品不同功能模块体验上的不满结果。

这个时候，产品团队就可以通过用户行为角度分组来具体分析存在哪些问题。还是以浏览器为例，新闻，小说，视频都应该是每日活跃的模块，如果做的好，应该表现出很高留存比例，如果分组分析中发现使用过某个功能的用户在之后的时间中留存情况很差，那就需要对这个功能进行优化了。同时，可以做竞品分析，看看相对于竞争对手，产品在哪些方面可以做进一步的优化和改进。

在本例中，我们按照新用户的注册时间分计算连续10天的留存率。

理想情况下，我们最终会得到一个这样的图表：



注：图源自网络，与本例无关

Step 1: 我们首先定义一些subquery来简化问题。New\_user\_activity将用户活动限制为新用户，cohort\_active\_user\_count计算每个每日群组中活跃用户的总数，也就是留存率计算中的分母：

with new\_user\_activity as(

select activity.\*

from activity

join user

on activity.DAY=user.JOINDATE

),

cohort\_active\_user\_count as(

select DAY, count(distinct UID) as count

from new\_user\_activity

group by DAY

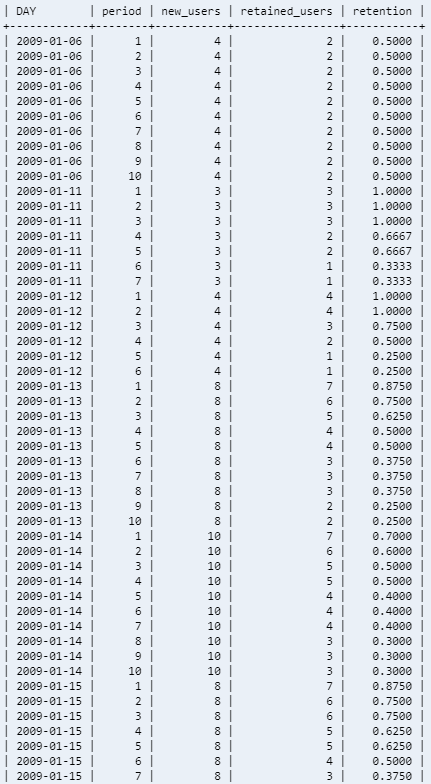
)

Step 2: 我们对主查询进行一些较小的更改。

计算留存期（period）:

Future\_activity.DAY-new\_user\_activity.DAY

通过留存期分组，我们可以清楚的计算每个cohort接下来的10天内的留存率变化。结果如图下所示：



Query如下：

select

DAY, period, new\_users, retained\_users, retention

from (

select

new\_user\_activity.DAY,

(future\_activity.DAY--new\_user\_activity.DAY) as period,

max(cohort\_size.count) as new\_users,

count(distinct future\_activity.UID) as retained\_users,

count(distinct future\_activity.UID)/max(cohort\_size.count) as retention

from

new\_user\_activity

left join

activity as future\_activity

on

new\_user\_activity.UID=future\_activity.UID

and

new\_user\_activity.DAY<future\_activity.DAY

and

new\_user\_activity.DAY>=subdate(future\_activity.DAY,10)

left join

cohort\_active\_user\_count as cohort\_size

on

new\_user\_activity.DAY=cohort\_size.DAY

group by

1,2

) tmp

where period is not null

order by DAY, period；

Step 3: 最后为了便于观察，我们将period转化为列。由于版本问题，我的SQL不支持pivot功能，这里我用case when函数为大家示例：

select

DAY, new\_users,

ifnull(max(case when period = 1 then retention end),0) DAY01,

ifnull(max(case when period = 2 then retention end),0) DAY02,

ifnull(max(case when period = 3 then retention end),0) DAY03,

ifnull(max(case when period = 4 then retention end),0) DAY04,

ifnull(max(case when period = 5 then retention end),0) DAY05,

ifnull(max(case when period = 6 then retention end),0) DAY06,

ifnull(max(case when period = 7 then retention end),0) DAY07,

ifnull(max(case when period = 8 then retention end),0) DAY08,

ifnull(max(case when period = 9 then retention end),0) DAY09,

ifnull(max(case when period = 10 then retention end),0) DAY10

from (

select new\_user\_activity.DAY,

(future\_activity.DAY--new\_user\_activity.DAY) as period,

max(cohort\_size.count) as new\_users,

count(distinct future\_activity.UID) as retained\_users,

count(distinct future\_activity.UID)/max(cohort\_size.count) as retention

from new\_user\_activity

left join activity as future\_activity

on new\_user\_activity.UID=future\_activity.UID

and new\_user\_activity.DAY<future\_activity.DAY

and new\_user\_activity.DAY>=subdate(future\_activity.DAY,10)

left join cohort\_active\_user\_count as cohort\_size

on new\_user\_activity.DAY=cohort\_size.DAY

group by 1,2

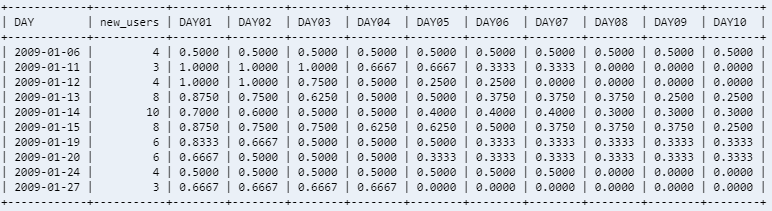
) tmp

where period is not null

group by DAY,

order by DAY;

好啦，这样我们的每日新增维度细分的 Cohort Analysis 表格就出来啦~



最后，我们也可以用该数据制作留存曲线图，分析比较不同cohort的用户的留存，这里就不多加说明。

**小结**

除了SQL，现在还有很多其他的计算留存的工具，操作简单，功能丰富，例如：Google Analytics, Tableau, 以及网站后台自带的数据分析软件等。对于实战来说，工具不是主要的，大家可以选择自己擅长的工具，最主要的是了解业务知识，要知其所以然，工具只是提供分析思路的一种方法。

“If you cannot measure it, you cannot improve it.”