

5.

use linkedwx;

drop table orders;

create table orders

(

num int not null primary key auto\_increment,

ordertime datetime not null,

customer char(50) not null

);

select \* from orders;

insert into orders (ordertime, customer) values ('2017-01-01 12:35:01', 'zhangsan');

insert into orders (ordertime, customer) values ('2017-01-02 12:35:01', 'zhangsan'), ('2017-01-02 12:40:01', 'lisi'),

('2017-01-02 13:40:01', 'lisi'), ('2017-01-03 12:40:01', 'lisi');

select date(t1.ordertime), count(distinct t1.customer) as total, count(distinct t2.customer) as old, count(distinct t1.customer) - count(distinct t2.customer) as newly

from orders as t1 left join orders as t2 on (t1.customer = t2.customer) and (date(t2.ordertime) < date(t1.ordertime))

group by date(t1.ordertime) order by date(t1.ordertime);

Part 1 – SQL

Given the below subset of a travel app’s schema, write executable SQL queries to answer the two questions below. Please answer in a single query and assume read-only access to the database (i.e. do not use CREATE TABLE).

Assume a PostgreSQL database, server timezone is UTC.

Table Name: trips

|  |  |
| --- | --- |
| Column Name: | Datatype: |
| id | integer |
| client\_id | integer (Foreign keyed to users.usersid) |
| driver\_id | integer (Foreign keyed to users.usersid) |
| city\_id | integer |
| client\_rating | integer |
| driver\_rating | integer |
| status | Enum(‘completed’, ‘cancelled\_by\_driver’, ‘cancelled\_by\_client’) |
| actual\_eta | integer |
| request\_at | timestamp with timezone |

Table Name: users

|  |  |
| --- | --- |
| Column Name: | Datatype: |
| usersid | integer |
| email | character varying |
| signup\_city\_id | integer |
| banned | Boolean |
| role | Enum(‘client’, ‘driver’, ‘partner’) |
| created\_at | timestamp with timezone |

1. Between Oct 1, 2013 at 10am PDT and Oct 22, 2013 at 5pm PDT, what percentage of requests made by unbanned clients each day were canceled in each city?
2. For city ids 1, 6, and 12, list the top three drivers by number of completed trips for each week between June 3, 2013 and June 24, 2013.

On a scale of 1-5 with 5 being for a perfect answer, where do you think your answer stands?

1.

select Request\_at as 'Day', City\_id as 'City',

(round(

sum(

case when status = 'cancelled\_by\_client' then 1

else 0 end) / count(\*)

,2)

) as 'Rate'

from trips join users on Client\_Id = Users\_Id

where Banned = 'No' and Role ='client' and

(Request\_at between (convert\_tz('2013-10-01 10:00:00','-07:00','+00:00')) and (convert\_tz ('2013-10-22 17:00:00','-07:00','+00:00')))

group by Request\_at, city\_id order by Request\_at asc

2.

select info.city\_id as 'City', info.users\_id as 'driver', info.week as 'week' from

(select city\_id, users\_id, date\_format(request\_at, '%Y-%u') as week, count(\*) as orders from trips join users on Driver\_Id = Users\_Id

where trips.status = 'completed' and City\_Id in (1,6,12)

and (Request\_at between '2013-06-03' and '2013-06-24')

group by date\_format(request\_at, '%Y-%u'), city\_id)

as info

where 3 >

(select count(distinct temp.users\_id) from

(select city\_id, users\_id, date\_format(request\_at, '%Y-%u') as week, count(\*) as orders from trips join users on Driver\_Id = Users\_Id

where trips.status = 'completed' and City\_Id in (1,6,12)

and (Request\_at between '2013-06-03' and '2013-06-24')

group by date\_format(request\_at, '%Y-%u'), city\_id)

as temp

where temp.city\_id = info.city\_id and temp.week = info.week and

temp.orders > info.orders)

**SQL查询每门课程的前几名**

<https://blog.csdn.net/wxnjob/article/details/43231491>

sql server查询每门课程的前两名的学生编号，课程编号，成绩并排序。

在SQL Server 数据库中，有一个表StudentAchievement（学生成绩）,

它有三个字段：StudentID(varchar(8),学生编号)，CourseID(varchar(10),课程编号)，Achievement(int ，成绩)，

写一条SQL语句，筛选出每门课程的前两名的学生编号，课程编号，成绩并排序。

--方法一

--将每科前两名学生的ID取出

/\*

这是个自连接问题，可以这样想，就是把studentachievement表看成两个表，A和B，

A取的是B表中筛选出来的ID，而B表中筛选出来的ID则要靠A中的courseID来筛选，

即A表里选出来的是和B表courseID相同的，但在B表中占前两项的值。

\*/

SELECT \*

FROM STUDENTACHIEVEMENT A

WHERE STUDENTID IN (SELECT TOP 2 STUDENTID

                       FROM STUDENTACHIEVEMENT B

                      WHERE B.COURSEID = A.COURSEID

                      ORDER BY B.ACHIEVEMENT DESC)

ORDER BY A.COURSEID, A.ACHIEVEMENT DESC

--方法二

--以科目，按成绩给记录编号，然后取出编号<=2的，就是每科前两名

SELECT \*

FROM STUDENTACHIEVEMENT A

WHERE (SELECT COUNT(\*)

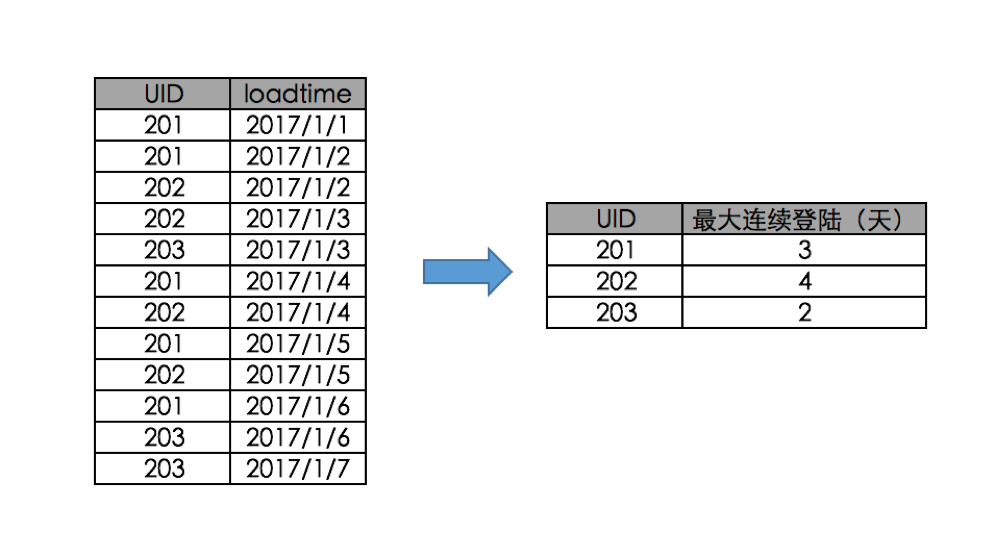
          FROM STUDENTACHIEVEMENT B

         WHERE B.COURSEID = A.COURSEID

           AND B.ACHIEVEMENT >= A.ACHIEVEMENT) <= 2

ORDER BY A.COURSEID, A.ACHIEVEMENT DESC

### SQL语句如何查询各个用户最长的连续登陆天数？

SQL语句如何查询各个用户最长的连续登陆天数？如图左边是源表User,右边是需要达到的查询结果，试了row\_number，但是无法达到连续登陆这个条件。

<https://bbs.csdn.net/topics/392243867>

Select UID,max(cnt) as cnt

From (

            Select UID,Grp\_No,count(\*) as cnt

            From (

                    Select UID,LoadTime,(Day(LoadTime)-ROW\_NUMBER() OVER (Partition By UID Order By UID,LoadTime)) as Grp\_No

                    From #Tmp\_Data

                  ) a

            Group By UID,Grp\_No

      ) a

Group By UID

**MYSQL version:**

use linkedwx;

show tables;

drop tables orders;

create table orders

(

num int not null primary key auto\_increment,

uid int not null,

loadtime datetime not null

);

select \* from orders;

(select \* from orders order by uid asc, loadtime desc);

insert into orders (uid, loadtime) values (201, '2017-01-01');

insert into orders (uid, loadtime) values (201, '2017-01-02'),

(202, '2017-01-02'), (202, '2017-01-03'), (203, '2017-01-03'),

(201, '2017-01-04'), (202, '2017-01-04'), (201, '2017-01-05'),

(202, '2017-01-05'), (201, '2017-01-06'), (203, '2017-01-06'),

(203, '2017-01-07');

insert into orders (uid, loadtime) values (204, '2017-01-02');

delete from orders where num = 14;

select uid, max(results) from

(select uid, (case when datediff(@cur, @cur := loadtime) = 1 and (@id - (@id := uid) = 0) then @count := @count + 1

else (@count := 1 and @id := uid) END) as results

from (select \* from orders order by uid asc, loadtime desc) as c, (select @cur := -1, @count := 0, @id := -1) as b) as temp

group by uid order by uid asc;

