Student Presentations
Lagging SQL

cs380
Dec 10

Lagging SQL

- Problem: how do you show the difference between two records
  - or simply how to you show parts of two “consecutive” records on the same line

- First problem — define consecutive
- Second problem — recognize consecutiveness
- Third problem — actually use 1 and 2.

The launch table

of the rocket database

- Question: how many days between launches
  - at a site?
  - of a vehicle?

- If I can do one, the other is easy

- 1: consecutive=next launch at same site (order by launchsite, date)

```
describe launch;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| Tag        | varchar(10)  | NO   | PRI | NULL    |       |
| JD         | varchar(12)  | NO   | PRI | NULL    |       |
| Date       | date         | YES  |     | NULL    |       |
| Vehicle    | varchar(20)  | YES  | MUL | NULL    |       |
| Flight     | varchar(20)  | YES  |     | NULL    |       |
| Mission    | varchar(30)  | YES  |     | NULL    |       |
| LaunchSite | varchar(10)  | YES  | MUL | NULL    |       |
| LaunchPad  | varchar(10)  | YES  |     | NULL    |       |
| Apogee     | mediumint(9) | YES  |     | NULL    |       |
| Category   | varchar(10)  | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
10 rows in set (0.001 sec)
```

```
select tag, date, vehicle, flight, launchsite from launch limit 2;
+----------+------------+---------+--------+------------+
| tag      | date       | vehicle | flight | launchsite |
+----------+------------+---------+--------+------------+
| 1942-A01 | 1942-06-13 | A-4     | 2      | HVP        |
| 1942-A02 | 1942-08-16 | A-4     | 3      | HVP        |
+----------+------------+---------+--------+------------+
2 rows in set (0.001 sec)
```

Consecutive records

- So getting a listing of consecutive records is easy enough.
  - Problem how to identify them
  - Even if there is an integer index
    - it may not be for the order you want
    - It cold have gaps

- Create an incrementing variable and increment it in the query.
  - watch for resetting the value!
  - watch for when the value increments too

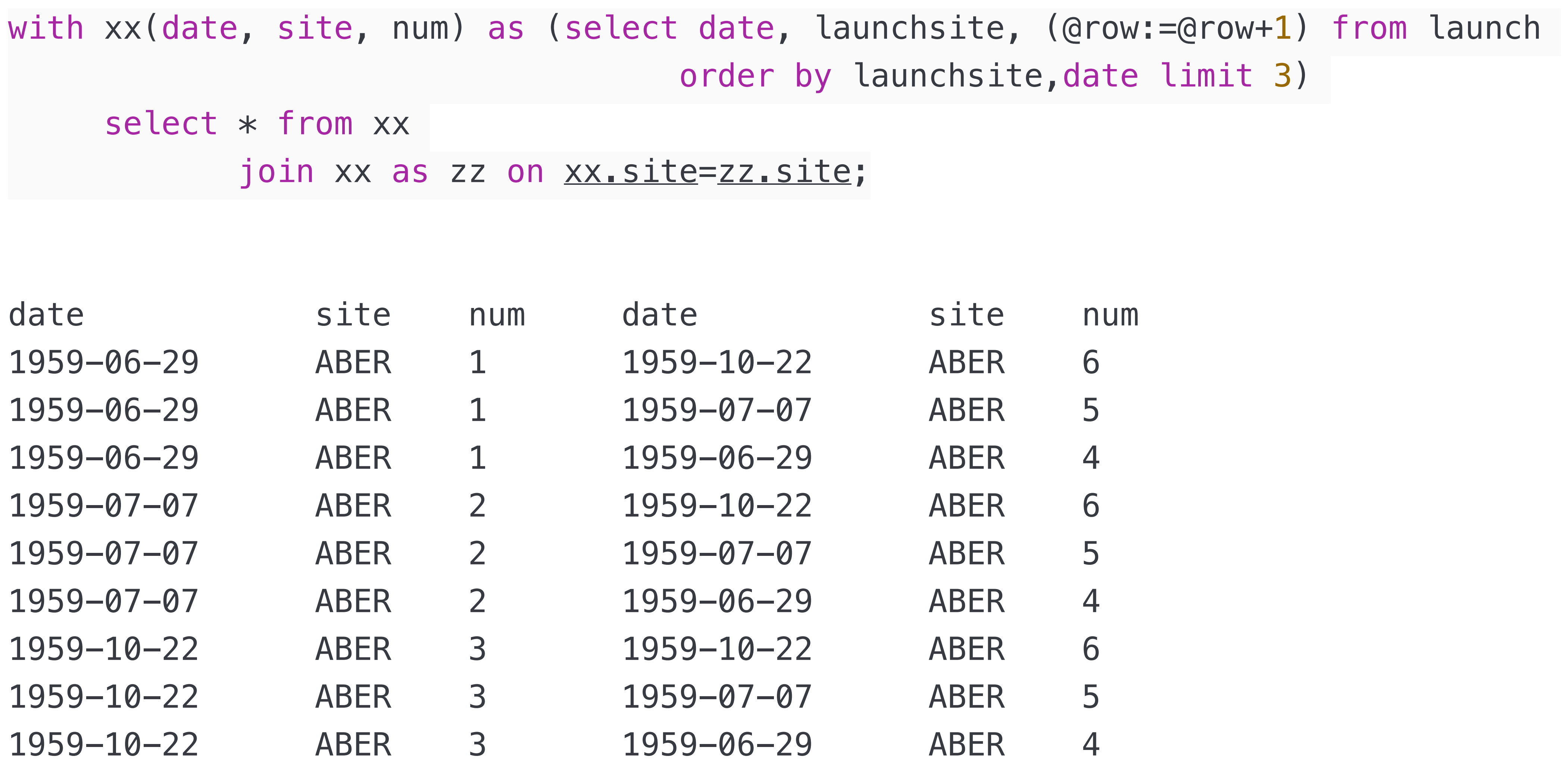
```
select date, launchsite from launch order by launchsite,date limit 5;
+------------+------------+
| date       | launchsite |
+------------+------------+
| 1959-06-29 | ABER       |
| 1959-07-07 | ABER       |
| 1959-10-22 | ABER       |
| 1960-01-02 | ABER       |
| 1960-01-07 | ABER       |
+-------+------------+
```

```
set @rowa:=0;
select date, launchsite, (@rowa:=@rowa+1) as rowid from launch
       order by launchsite,date limit 5;
+------------+------------+-------+
| date       | launchsite | rowid |
+------------+------------+-------+
| 1959-06-29 | ABER       |     1 |
| 1959-07-07 | ABER       |     2 |
| 1959-10-22 | ABER       |     3 |
| 1960-01-02 | ABER       |     4 |
| 1960-01-07 | ABER       |     5 |
+------------+------------+-------+
5 rows in set (0.027 sec)
```

```
select date, launchsite, (@rowa:=@rowa+1) as rowid from launch
       order by launchsite,date limit 5;
+------------+------------+-------+
| date       | launchsite | rowid |
+------------+------------+-------+
| 1959-06-29 | ABER       |     6 |
| 1959-07-07 | ABER       |     7 |
| 1959-10-22 | ABER       |     8 |
| 1960-01-02 | ABER       |     9 |
| 1960-01-07 | ABER       |    10 |
+------------+------------+-------+
5 rows in set (0.027 sec)
```

Idea: self join!

- Create a set that I want (use with).
  - Join it to itself!
  - Almost, but the value of num incremented
    - With acts like a store procedure so it only gets expanded when required.
      - It is required twice!
        - So the value of row is computed twice.
        - Cannot reset to zero every time
          - (maybe could but I do not know how)



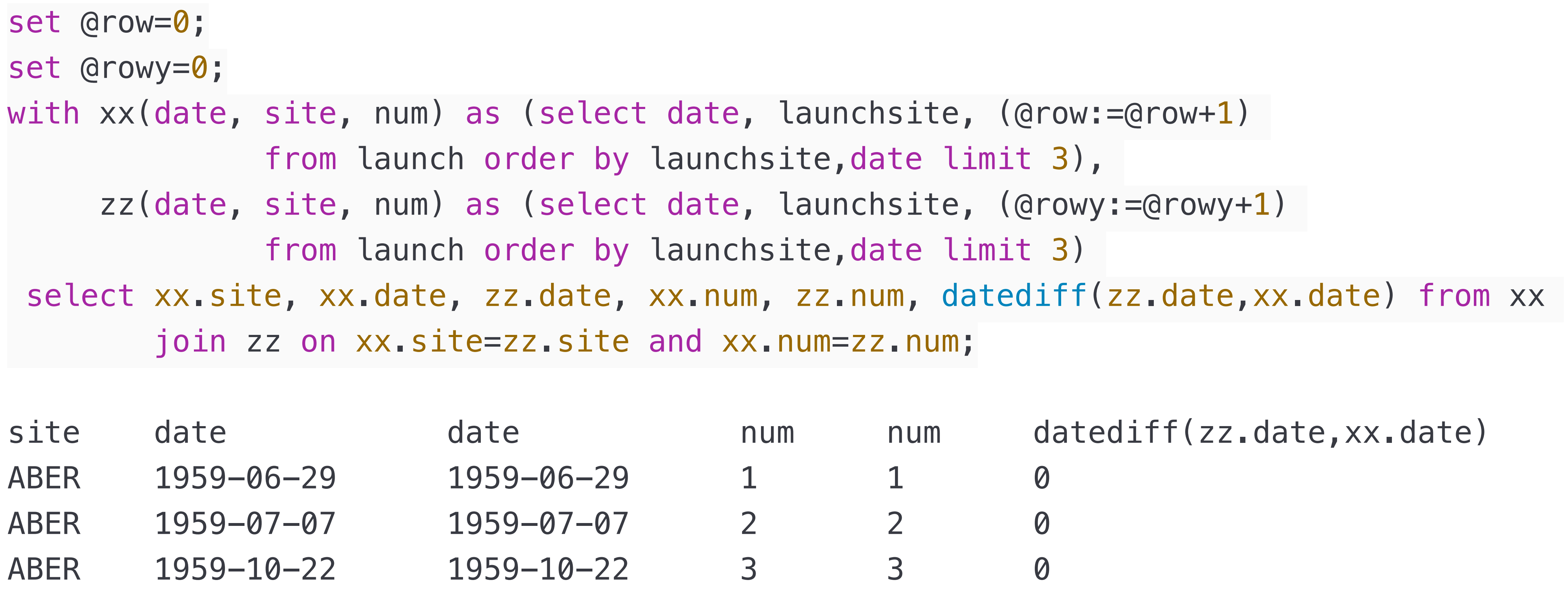
```
with xx(date, site, num) as (select date, launchsite, (@row:=@row+1) from launch
                                   order by launchsite,date limit 3)
     select * from xx
            join xx as zz on xx.site=zz.site;
```

```
date            site    num     date            site    num
1959-06-29      ABER    1       1959-10-22      ABER    6
1959-06-29      ABER    1       1959-07-07      ABER    5
1959-06-29      ABER    1       1959-06-29      ABER    4
1959-07-07      ABER    2       1959-10-22      ABER    6
1959-07-07      ABER    2       1959-07-07      ABER    5
1959-07-07      ABER    2       1959-06-29      ABER    4
1959-10-22      ABER    3       1959-10-22      ABER    6
1959-10-22      ABER    3       1959-07-07      ABER    5
1959-10-22      ABER    3       1959-06-29      ABER    4
```

Make two explicit subsets

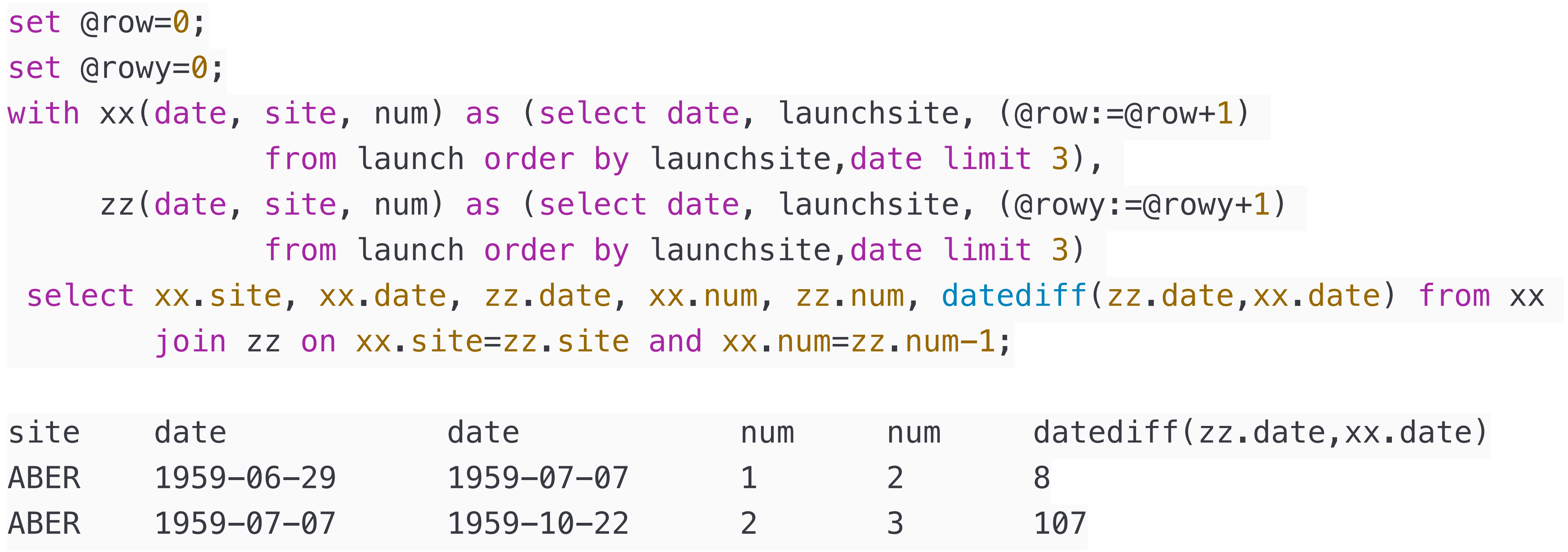
- Need another variable but otherwise easy.
    - That works
- Now to get that offset
    - Just subtract 1

- Small(ish) problem efficiency
    - get rid of “limit 3”
    - On 66000 records this takes 18 seconds!
        - Theory: string comparisons are slow
            - eliminate “xx.site=zz.site" from join
                - 160 seconds
            - String comp is not issue!
        - Theory: “row” comparison is the issue
            - Without row comparison the join creates a lot of rows
                - next page
            - replace “row” comparison with date comparison
                - 1.6 seconds
        - Theory: subtraction in join is the issue
            - without subtraction 0.8 seconds!
        - **Subtraction was the whole point!**



```
set @row=0;
set @rowy=0;
with xx(date, site, num) as (select date, launchsite, (@row:=@row+1)
              from launch order by launchsite,date limit 3),
     zz(date, site, num) as (select date, launchsite, (@rowy:=@rowy+1)
              from launch order by launchsite,date limit 3)
 select xx.site, xx.date, zz.date, xx.num, zz.num, datediff(zz.date,xx.date) from xx
        join zz on xx.site=zz.site and xx.num=zz.num;
```

```
site    date            date            num     num     datediff(zz.date,xx.date)
ABER    1959-06-29      1959-06-29      1       1       0
ABER    1959-07-07      1959-07-07      2       2       0
ABER    1959-10-22      1959-10-22      3       3       0
```



```
set @row=0;
set @rowy=0;
with xx(date, site, num) as (select date, launchsite, (@row:=@row+1)
              from launch order by launchsite,date limit 3),
     zz(date, site, num) as (select date, launchsite, (@rowy:=@rowy+1)
              from launch order by launchsite,date limit 3)
 select xx.site, xx.date, zz.date, xx.num, zz.num, datediff(zz.date,xx.date) from xx
        join zz on xx.site=zz.site and xx.num=zz.num-1;
```

```
site    date            date            num     num     datediff(zz.date,xx.date)
ABER    1959-06-29      1959-07-07      1       2       8
ABER    1959-07-07      1959-10-22      2       3       107
```

How many rows?

- Each of the xx and zz sets contains 63688 rows
  - so max rows from join is 63688^2
    - 4056161344
    - This would happen if only 1 site
- Actual number is sum of square of number at each site.
  - How to do this using only sql????
  - Honestly, I would be very tempted to use python and sql....

```
 # This join will create a LOT of rows — but how many
select xx.site, xx.date, zz.date, xx.num, zz.num,
        datediff(zz.date,xx.date) from xx
   join zz on xx.site=zz.site;

# number of rows in the table
select count(*) from launch;
      63688

# This is the max possible
select count(*) * count(*) from launch;
      4056161344
```

```
#Now to compute actual number
# aa query gets the count at each site
# bb adds everything up, but has a lot of rows
# final select just uses the max from bb
set @qq:=0;
set @rr:=0;
with aa(cc) as (select count(*) from launch group by launchsite),
     bb(mm,nn,oo) as (select cc, @rr:=@rr+cc, @qq:=@qq+cc*cc from aa)
   select max(oo), max(oo)/(max(nn)*max(nn)) from bb;

max(oo)         max(oo)/(max(nn)*max(nn))
168112092       0.0414
```

About 4% of the possible so still better than cross-product

This is a actual number of rows that the query would create

Row numbering by group

Previous slide just got total in group

- sql has a "rank" function which should do much the same thing,
    - it is unreliable/useless
        - My tests, the total is correct but replications along the way

```
set @pname:='xxxx';
set @rank:=1;
select launchsite,
        @rank:=if(@pname=launchsite, @rank+1,
                                    if(@pname:=launchsite,1,1))
       from launch
       order by launchsite, date;
…
YSNYA   84
YSNYA   85
YSNYA   86
YUK     1
YUMA    1
YUMA    2
YUMA    3
YUMA    4
YUMA    5
…

# Equivalent to above, just avoids separate "set"
select launchsite,
       @rank:=if(@pname=launchsite, @rank+1,
                                   if(@pname:=launchsite,1,1))
       from launch as ll,
            (select @pname:='yweruiyw') as pp,
            (select @rank:=1) as rr
       order by launchsite limit 10;
```

Doing full cross-product, but there is only one row in two of these

Naming required when there is more than one part of "from"

Efficiency

- Just start one counter before the other!

- several possibly slow operations
  - two selects
  - join
- Flexible — easily change offset
- Awkward — requires two separate selects
- Readable

```
set @row=0;
set @rowy=-1;
with xx(date, site, num) as (select date, launchsite, (@row:=@row+1)
              from launch order by launchsite,date limit 3),
     zz(date, site, num) as (select date, launchsite, (@rowy:=@rowy+1)
              from launch order by launchsite,date limit 3)
 select xx.site, xx.date, zz.date, xx.num, zz.num, datediff(zz.date,xx.date) from xx
        join zz on xx.site=zz.site and xx.num=zz.num;
```

```
site    date            date            num     num     datediff(zz.date,xx.date)
ABER    1959-06-29      1959-07-07      1       2       8
ABER    1959-07-07      1959-10-22      2       3       107
```

Use lagging variables

- Idea use variables that hold the value from the prior row
- Note that @psite is “reported” before it is updated
  - same for @pdate

- Fast: less than 40% time of previous
- Awkward:
  - lag of 1 is OK.
  - 2 would be bad, 5 awful
- Undefined
  - mysql does not guarantee the order of evaluations in select

```
set @psite='xgxgxg';
set @pdate=curdate();
with aa(psite, site, pdate, date) as
     (select @psite, @psite:=launchsite, @pdate, @pdate:=date
             from launch order by launchsite, date)
  select site, date, pdate, datediff(date, pdate) from aa where site=psite;
```

```
site    date    pdate   datediff(date, pdate)
ABER    1959-07-07      1959-06-29      8
ABER    1959-10-22      1959-07-07      107
ABER    1960-01-02      1959-10-22      72
```

Use lag function

- LAG(XXX,n) OVER (PARTITION BY yyy ORDER BY zzz)
    - XXX==the column to lag
    - n==the amount of lag
    - over — set conditions on lag
        - PARTITION BY yyy
            - grouping
        - order by zzz
            - sorting
    - In prior queries we got partition by and order by using 2 keys on “order by”.
        - LAG is independent of “order by”

```
select launchsite, date, datediff(date, lag(date,1)
                                        over (partition by launchsite order by date))
    from launch
    order by launchsite,date

launchsite    date        diff
ABER    1959-06-29        NULL
ABER    1959-07-07        8
ABER    1959-10-22        107
```

```
with aa(site, date, diff) as (
       select launchsite, date,
             datediff(date, lag(date,1) over (partition by launchsite order by date))
          from launch order by launchsite,date limit 3
   )
   select * from aa where diff is not NULL;
launchsite    date        diff
ABER    1959-07-07        8
ABER    1959-10-22        107
```