# Packing Date Intervals

By Phil, sort of

# Okay, what is this?

Opioid Abuse

Aim 2 of the project asks that we investigate whether or not certain prescribing guidelines are maintained for Opioid Prescriptions e.g.

1. Are Urine Tox Screens ordered around initiation of an opioid
2. Not co-prescribing Opioids and BZDs

In order to do this, we need to create “treatment windows” that is:

A time interval during which the patient is said to have been prescribed Opioids.

# Okay cool. That doesn’t sound so hard.

It is.

First of all, sometimes patients are prescribed multiple opioids per period or just have overlapping prescription dates for a variety of reasons (abuse or otherwise).

There is often a period “in between” prescriptions because patients forget to request a refill, there was a weekend etc etc which would break up these windows. This doesn’t really reflect the “concept” of a treatment window though. Just because you ran out of meds for a week because of an insurance issue or something doesn’t really mean you weren’t “prescribed” opioids as part of ongoing treatment.

# Oh I see.

Yes.

So we need to identify and combine both:

1. Overlapping Opioid Windows
2. Give a ‘grace period’ for what constitutes a contiguous “Prescription Window”

# Example

This is a sample of prescriptions from a single patient (it cuts off at the bottom, there are > 150 more prescriptions).

Here we see two examples of our test cases where there is overlap of dates (red) and where there is no overlap but the dates are very close (blue)

| FillDateTime | DaysSupply | EndDate |
|---|---|---|
| 2011-02-04 00:00:00 | 30 | 2011-03-06 00:00:00.000 |
| 2011-02-24 00:00:00 | 30 | 2011-03-26 00:00:00.000 |
| 2011-03-26 00:00:00 | 30 | 2011-04-25 00:00:00.000 |
| 2011-04-25 00:00:00 | 30 | 2011-05-25 00:00:00.000 |
| 2011-05-31 00:00:00 | 30 | 2011-06-30 00:00:00.000 |
| 2011-06-24 00:00:00 | 30 | 2011-07-24 00:00:00.000 |
| 2011-08-01 00:00:00 | 30 | 2011-08-31 00:00:00.000 |
| 2011-08-21 00:00:00 | 30 | 2011-09-20 00:00:00.000 |
| 2011-09-13 00:00:00 | 30 | 2011-10-13 00:00:00.000 |
| 2011-10-04 00:00:00 | 30 | 2011-11-03 00:00:00.000 |
| 2011-11-02 00:00:00 | 30 | 2011-12-02 00:00:00.000 |
| 2011-12-02 00:00:00 | 30 | 2012-01-01 00:00:00.000 |
| 2012-01-01 00:00:00 | 30 | 2012-01-31 00:00:00.000 |
| 2012-01-31 00:00:00 | 30 | 2012-03-01 00:00:00.000 |
| 2012-02-23 00:00:00 | 30 | 2012-03-24 00:00:00.000 |
| 2012-03-14 00:00:00 | 30 | 2012-04-13 00:00:00.000 |
| 2012-04-13 00:00:00 | 30 | 2012-05-13 00:00:00.000 |
| 2012-05-02 00:00:00 | 15 | 2012-05-17 00:00:00.000 |
| 2012-05-13 00:00:00 | 30 | 2012-06-12 00:00:00.000 |
| 2012-05-31 00:00:00 | 15 | 2012-06-15 00:00:00.000 |
| 2012-06-12 00:00:00 | 30 | 2012-07-12 00:00:00.000 |
| 2012-06-22 00:00:00 | 15 | 2012-07-07 00:00:00.000 |
| 2012-07-12 00:00:00 | 30 | 2012-08-11 00:00:00.000 |
| 2012-07-13 00:00:00 | 15 | 2012-07-28 00:00:00.000 |
| 2012-08-02 00:00:00 | 13 | 2012-08-15 00:00:00.000 |
| 2012-08-07 00:00:00 | 30 | 2012-09-06 00:00:00.000 |
| 2012-08-15 00:00:00 | 19 | 2012-09-03 00:00:00.000 |
| 2012-09-05 00:00:00 | 28 | 2012-10-03 00:00:00.000 |
| 2012-10-04 00:00:00 | 28 | 2012-11-01 00:00:00.000 |
| 2012-10-26 00:00:00 | 28 | 2012-11-23 00:00:00.000 |
| 2012-11-21 00:00:00 | 28 | 2012-12-19 00:00:00.000 |

# Okay cool. Do you have any solutions?

Sure.

My first instinct was to explore the entire thing out of SQL server and do a python/pandas thing to do because I am more familiar with that. Something probably to do with iterating over rows and comparing dates.

But, it would probably be faster and less annoying to just get it done in SQL, so that is what I endeavored to do.

My solution is on the next slide.

# Solution

That is the solution. It works. The output is shown at the bottom.

I only really understand about half of what is going on here. There are two Window Functions here. The first one aggregates via a MAX function partitioned over Patients, and then flags dates that overlap with the next date. The second assigns group names to intervals. The final query groups the rows by patient and interval and finds the Max and Min.

The arrow points to the place where you can change the “grace period” to allow for gaps between windows. To simply find overlaps, you can just exclude that entire DATEADD function.

But this isn’t really the point of my presentation.

```
WITH C1 as (
    SELECT
        *,
        CASE WHEN
            DATEADD(Day, -7, FillDateTime) <= MAX(EndDate) OVER(PARTITION BY PatientICN
                                                                ORDER BY FillDateTime, EndDate
                                                                ROWS BETWEEN UNBOUNDED PRECEDING AND 1 PRECEDING)
            THEN 0 ELSE 1
        END as isstart
    FROM Dflt.PM_Opioid_Windows
),

C2 as (
    SELECT
        *,
        SUM(isstart) OVER(PARTITION BY PatientICN
                          ORDER BY FillDateTime, EndDate
                          ROWS UNBOUNDED PRECEDING) as grp
    FROM C1
)

SELECT
    PatientICN,
    MIN(FillDateTime) as StartTime,
    MAX(EndDate) as EndDate
FROM C2

GROUP BY PatientICN, grp
ORDER BY StartTime
```

| StartTime | EndDate |
|---|---|
| 2011-02-04 00:00:00 | 2011-07-24 00:00:00.000 |
| 2011-08-01 00:00:00 | 2017-04-04 00:00:00.000 |
| 2022-06-10 00:00:00 | 2022-07-10 00:00:00.000 |

I had previously tried a few other ways, joining the table onto itself and doing pairwise comparisons between dates but this either didn't work (because I am bad at SQL) or ran so long that Mr. VINCI turned off the query. (Actually, since I was running this over the weekend, I think it may have had something to do with RB02 maintenance but that's speculation). Regardless, it would have probably taken a long time to run. North of a few hours.

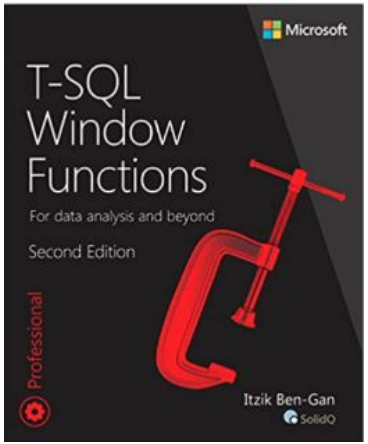
In contrast, this Query takes a grand total of 04.390 seconds, reducing a table of 1.5M rows to 559K containing 97K separate patients.

# Okay. So.

I did not come up with that query. I found it in a book. I book that I think many of us might find quite useful. I've only just started pursuing it but it looks quite good.

The Book is called:

**T-SQL Window Functions: For data analysis and beyond (Developer Reference) 2nd Edition by Itzak Ben-Gan**



# Fin

I'll of course be putting the script onto the GitHub for future reference. I am confident this sort of thing comes up often.