Paper 341-2019

Time Since Last Dose: Anatomy of a SQL Query

Derek Morgan, PAREXEL International

ABSTRACT

Even though much of what we need to accomplish as statistical programmers can be accomplished using the DATA step, SQL can provide an alternative to large amounts of data manipulation. One such case is determining time since last (or first) dose. This paper will walk you through a SQL solution in place of multiple SORTs, MERGEs, TRANSPOSEs, and use of the LAG function. Not only is the code economical, but the execution is also economical in terms of execution. This may also help to increase your understanding of how the WHERE statement, and the SQL GROUP BY, and HAVING clauses work.

INTRODUCTION

Before we begin to dissect the code, think of this as a recipe to calculate the time since last dose. Every recipe needs an ingredient list, and this is no different. What you will need to use this method:

- Adverse event start date/time as standard variables (ASTDT, ASTDTM).
- Adverse event subject identifier (USUBJID).
- Adverse event sequence numbers (AESEQ) – I'll discuss why this is an important variable in a few moments.
- Exposure start/end dates/times as standard SAS date variables (EXSTDT, EXSTDTM, EXENDT, EXENDTM).
- Exposure sequence number (EXSEQ).

The dates and times will be used for comparison in the query to get your ultimate result. The AE subject identifier and AE sequence number will allow you to merge your result into the ADAE ADaM dataset, as the time to event is unique to each subject's adverse event, and these two variables will uniquely identify any single record in the ADAE dataset. The exposure sequence number is used for quality control. While it is possible to include other AE variables and eliminate the step to merge this result to your ADAE dataset, the additional time involved in moving all the extraneous data would be like requiring an entire meal be cooked before you can work on this one dish.

THE DATA

For this paper, we will be using the following data to demonstrate the how the query works. Each dataset only contains the variables necessary for the query; they are not complete ADaM datasets.

This hypothetical study runs for 10 weeks, and dosing is scheduled weekly. A subject can miss one dose without being terminated. In our sample data, subject 0001 has missed a dose between weeks 5 and 7 (indicated by ***bold italics***.) If this dataset were to include records for missed exposures, EXSEQ would not correspond with the actual number of the exposure for each subject. Our sample exposure data is shown below in Sample Data 1:

usubjid	exseq	exstdtc	exstdt	exstdtm
0001	1	2018-12-01T00:07:00	01DEC2018	01DEC2018:00:07:00
0001	2	2018-12-08T00:02:00	08DEC2018	08DEC2018:00:02:00
0001	3	2018-12-15T00:06:00	15DEC2018	15DEC2018:00:06:00
0001	4	2018-12-22T00:10:00	22DEC2018	22DEC2018:00:10:00
0001	5	2018-12-29T00:09:00	29DEC2018	29DEC2018:00:09:00
0001	***6***	***2019-01-12T00:01:00***	***12JAN2019***	***12JAN2019:00:01:00***
0001	7	2019-01-19T00:04:00	19JAN2019	19JAN2019:00:04:00
0001	8	2019-01-26T00:07:00	26JAN2019	26JAN2019:00:07:00
0001	9	2019-02-02T00:05:00	02FEB2019	02FEB2019:00:05:00
0002	1	2019-03-21T00:06:00	21MAR2019	21MAR2019:00:06:00
0002	2	2019-03-28T00:04:00	28MAR2019	28MAR2019:00:04:00
0002	3	2019-04-04T00:05:00	04APR2019	04APR2019:00:05:00
0002	4	2019-04-11T00:10:00	11APR2019	11APR2019:00:10:00
0002	5	2019-04-18T00:02:00	18APR2019	18APR2019:00:02:00
0002	6	2019-04-25T00:01:00	25APR2019	25APR2019:00:01:00
0002	7	2019-05-02T00:07:00	02MAY2019	02MAY2019:00:07:00
0002	8	2019-05-09T00:02:00	09MAY2019	09MAY2019:00:02:00
0002	9	2019-05-16T00:01:00	16MAY2019	16MAY2019:00:01:00
0002	10	2019-05-23T00:02:00	23MAY2019	23MAY2019:00:02:00

Sample Data 1: Exposure Data (PSDATA.EX)

The adverse event data may or may not include a time of event, although all the adverse events in the table have a complete date. This method requires a complete date at the least; therefore, in the real world, you would use the date variable after imputation of any partial dates. Sample Data 2 follows:

usubjid	aeseq	aestdtc	astdt	astdtm
0001	1	2018-12-18	18DEC2018	.
0001	2	2018-12-13	13DEC2018	.
0001	3	2018-12-09	09DEC2018	.
0001	4	2019-01-08	08JAN2019	.
0001	5	2019-02-09T05:54	09FEB2019	09FEB2019:05:54:00
0001	6	2019-01-11T11:35	11JAN2019	11JAN2019:11:35:00
0002	1	2019-05-08T08:30	08MAY2019	08MAY2019:08:30:00
0002	2	2019-05-18T09:28	18MAY2019	18MAY2019:09:28:00
0002	3	2019-05-05T03:19	05MAY2019	05MAY2019:03:19:00

Sample Data 2: Adverse Event Data (PSDATA.AE)

STEP 1: USING SQL TO JOIN THE EX AND AE DATASETS

A simple equijoin (Sample Code 1) will create a dataset with 84 records, one record for each combination of AE and exposure by subject (9 exposures for subject 0001×6 adverse events, and 10 exposures for subject 0002×3 adverse events.) The entire equijoin dataset is in Appendix 1. The variable LASTDOSE is calculated for each record:

```
PROC SQL;
CREATE TABLE basic_join AS
SELECT DISTINCT ae.usubjid, aeseq, astdt, astdtm, COUNT(ex.usubjid) AS expnum, exstdt,
exstdtm, astdt - exstdt AS lastdose 'Days Since Last Dose', exseq
FROM psdata.ae ae, psdata.ex ex
WHERE ae.usubjid EQ ex.usubjid
ORDER BY ae.usubjid, aeseq
;
QUIT;
```

Sample Code 1: Simple Equijoin of EX and AE Datasets

Table 1 displays the records from the equijoin for each subject's first adverse event. The variable LASTDOSE is calculated for each exposure. To simplify, only the dates are shown.

usubjid	aeseq	exseq	astdt	exstdt	lastdose
0001	1	1	18DEC2018	01DEC2018	17
0001	1	2	18DEC2018	08DEC2018	10
0001	***1***	***3***	***18DEC2018***	***15DEC2018***	***3***
0001	1	4	18DEC2018	22DEC2018	-4
0001	1	5	18DEC2018	29DEC2018	-11
0001	1	6	18DEC2018	12JAN2019	-25
0001	1	7	18DEC2018	19JAN2019	-32
0001	1	8	18DEC2018	26JAN2019	-39
0001	1	9	18DEC2018	02FEB2019	-46
0002	1	1	08MAY2019	21MAR2019	48
0002	1	2	08MAY2019	28MAR2019	41
0002	1	3	08MAY2019	04APR2019	34
0002	1	4	08MAY2019	11APR2019	27
0002	1	5	08MAY2019	18APR2019	20
0002	1	6	08MAY2019	25APR2019	13
0002	***1***	***7***	***08MAY2019***	***02MAY2019***	***6***
0002	1	8	08MAY2019	09MAY2019	-1
0002	1	9	08MAY2019	16MAY2019	-8
0002	1	10	08MAY2019	23MAY2019	-15

Table 1: Equijoin Result for Each Subject's First Adverse Event

The most recent dosing for each subject's adverse event is shown in Table 1 with ***bold italics***. However, obtaining these specific records from this equijoin dataset is a just a filtering process. In this simplified state where only dates are taken into account, the most recent dose is defined as the minimum value of LASTDOSE is greater than zero. If LASTDOSE is negative, then the event occurred prior to dosing. Any values of LASTDOSE greater than the minimum indicates that there are prior doses after the event. PROC SQL has done the heavy lifting, all you have to do is filter.

How do you filter? If you don't quite trust PROC SQL, you could take this dataset and run it through a DATA step. However, that's additional execution overhead, and the sequential record processing of the DATA step takes much more time, relatively speaking. Let's create our filter in the same SQL step.

FILTERING IN SQL

With SQL, you can consider the WHERE clause as a filter. Even the equijoin used in Sample Code 1 is a filter. Without it, PROC SQL step would have created a dataset with 171 records! The WHERE clause in Sample Code 1 causes SQL to remove records that do not satisfy the condition, leaving the 84 records in Appendix 1. We can enhance the WHERE clause to remove any records where the exposure occurs after the event date. Since this is an additional condition, it is connected with AND. The additional condition to accomplish this is highlighted in Sample Code 2:

```
WHERE(ae.usubjid EQ ex.usubjid AND (exstdt LT astdt OR (exstdt EQ astdt AND (astdtm GE
exstdtm OR astdtm IS MISSING)))
```

Sample Code 2: Enhancing the WHERE Clause

We compare the dates (EXSTDT is less than ASTDT.) However, because both exposures and adverse events may have a time associated with the date, we also need to check the datetime of dosing against the datetime of the event, but only for events that occur on the day of dosing. If the event has no date or time, we also want to consider those records as well. We are now down to 49 records (Appendix 2Appendix 1) from our original 84. Table 2 shows the result of our enhanced equijoin for each subject's first adverse event. As you can see, any record from Table 1 with a negative value for LASTDOSE has been removed:

usubjid	aeseq	exseq	astdt	astdtm	exstdt	exstdtm	lastdose
0001	1	1	18DEC2018	.	01DEC2018	01DEC2018:00:07:00	17
0001	1	2	18DEC2018	.	08DEC2018	08DEC2018:00:02:00	10
0001	1	3	18DEC2018	.	15DEC2018	15DEC2018:00:06:00	3
0002	1	1	08MAY2019	08MAY2019:08:30:00	21MAR2019	21MAR2019:00:06:00	48
0002	1	2	08MAY2019	08MAY2019:08:30:00	28MAR2019	28MAR2019:00:04:00	41
0002	1	3	08MAY2019	08MAY2019:08:30:00	04APR2019	04APR2019:00:05:00	34
0002	1	4	08MAY2019	08MAY2019:08:30:00	11APR2019	11APR2019:00:10:00	27
0002	1	5	08MAY2019	08MAY2019:08:30:00	18APR2019	18APR2019:00:02:00	20
0002	1	6	08MAY2019	08MAY2019:08:30:00	25APR2019	25APR2019:00:01:00	13
0002	1	7	08MAY2019	08MAY2019:08:30:00	02MAY2019	02MAY2019:00:07:00	6

Table 2: Enhanced Equijoin Result for Each Subject's First Adverse Event

THE HAVING CLAUSE

Now we need to select the correct record from this list. We want to select the most recent dose from it, which is the maximum value of EXSTDTM. We can't use summary functions in a WHERE clause (Example 1.) Therefore, we will use the HAVING clause, which does allow the use of summary functions:

```
PROC SQL;
CREATE TABLE bad_where AS
SELECT ae.usubjid, aeseq, astdt, astdtm, COUNT(ex.usubjid) AS expnum, exstdt, exstdtm,
       astdt - exstdt AS lastdose 'Days Since Last Dose', exseq
FROM psdata.ae ae, psdata.ex ex
WHERE((ae.usubjid EQ ex.usubjid AND (exstdt LT astdt OR (exstdt EQ astdt AND (astdtm
GE exstdtm OR astdtm IS MISSING)))) AND MAX(exsdtm) eq exsdtm)
ORDER BY ae.usubjid, exseq, aeseq
;
QUIT;
```

ERROR: Summary functions are restricted to the SELECT and HAVING clauses only.

Example 1: Trying to Use a Summary Function in a WHERE Clause

Example 1 demonstrates what happens if you try to use a summary function in a WHERE clause. If you are going to use a summary function, it has to be done in a HAVING clause. When you use the HAVING clause, it applies to the result of the SQL query after the WHERE clause has been applied. The HAVING clause is generally used in conjunction with a GROUP BY clause. In fact, using the HAVING clause by itself will likely lead to an unexpected result as shown in Example 2:

```
PROC SQL;
CREATE TABLE just_having AS
SELECT DISTINCT ae.usubjid, aeseq, astdt, astdtm, COUNT(ex.usubjid) AS expnum, exstdt,
exstdtm, astdt - exstdt AS lastdose 'Days Since Last Dose', exseq
FROM psdata.ae ae, psdata.ex ex
WHERE(ae.usubjid EQ ex.usubjid AND (exstdt LT astdt OR (exstdt EQ astdt AND (astdtm GE
exstdtm OR astdtm IS MISSING)))
HAVING MAX(exstdtm) EQ exstdtm
ORDER BY ae.usubjid, aeseq
;
QUIT;
```

usubjid	aeseq	exseq	astdt	astdtm	exstdt	exstdtm	lastdose
0002	**2**	9	18MAY2019	18MAY2019:09:28:00	16MAY2019	16MAY2019:00:01:00	2

Example 2: Using the HAVING Clause by Itself

What happened here? Why is only one record selected? As noted before, the HAVING clause applies to the result of the query after the WHERE clause has been executed, and the HAVING clause asked for the maximum value of the dosing datetime from that result. Unless you have multiple records with the same maximum datetime, you will only get one record. For our data, this is exposure #9, adverse event #2 for subject 0002. You can validate this by looking at Appendix 2.

THE GROUP BY CLAUSE

We don't just want the maximum value of EXSTDT, we want it for each adverse event for each subject. The GROUP BY clause allows you to perform any calculations in the SELECT or HAVING clauses within BY groups. In this case, it applies to both (the COUNT() function in the SELECT, and the MAX() function in the HAVING clause. This leads to…

THE FINISHED PRODUCT

The code itself is simple, and given in its entirety as Sample Code 3. You can paste this code and change the appropriate variable names for your data:

```
PROC SQL;
CREATE TABLE most_recent_exp AS
SELECT DISTINCT ae.usubjid, aeseq, astdt, astdtm, COUNT(ex.usubjid) AS expnum, exstdt,
exstdtm, astdt - exstdt AS lastdose 'Days Since Last Dose', exseq
FROM psdata.ae ae, psdata.ex ex
WHERE(ae.usubjid EQ ex.usubjid AND (exstdt LT astdt OR (exstdt EQ astdt AND (astdtm GE
exstdtm OR astdtm IS MISSING)))
GROUP BY ae.usubjid, aeseq
HAVING MAX(exstdtm) EQ exstdtm
ORDER BY ae.usubjid, aeseq
;
QUIT;
```

Sample Code 3: An SQL Query to Obtain the Most Recent Exposure for Adverse Events

Our result is below in Table 3.

usubjid	aeseq	exseq	astdt	astdtm	exstdt	exstdtm	lastdose
0001	**1**	3	18DEC2018	.	15DEC2018	15DEC2018:00:06:00	3
0001	**2**	2	13DEC2018	.	08DEC2018	08DEC2018:00:02:00	5
0001	**3**	2	09DEC2018	.	08DEC2018	08DEC2018:00:02:00	1
0001	**4**	5	08JAN2019	.	29DEC2018	29DEC2018:00:09:00	10
0001	**5**	9	09FEB2019	09FEB2019:05:54:00	02FEB2019	02FEB2019:00:05:00	7
0001	**6**	5	11JAN2019	11JAN2019:11:35:00	29DEC2018	29DEC2018:00:09:00	13
0002	**1**	7	08MAY2019	08MAY2019:08:30:00	02MAY2019	02MAY2019:00:07:00	6
0002	**2**	9	18MAY2019	18MAY2019:09:28:00	16MAY2019	16MAY2019:00:01:00	2
0002	**3**	7	05MAY2019	05MAY2019:03:19:00	02MAY2019	02MAY2019:00:07:00	3

Table 3: Most Recent Dose from Event, by Subject

The most recent exposure has been selected for each adverse event. The additional calculation (EXPNUM) to count the number of actual exposures is not shown here, because the actual exposure number is the same as EXSEQ. EXPNUM would be different from EXSEQ if there were records for missed exposures in the exposure dataset.

CONCLUSION

This is the beauty of using SQL to obtain these results. A few lines of code take the place of multiple traditional DATA steps. We started with 171 records in a Cartesian product (not shown, but you can run the code without the WHERE, GROUP BY and HAVING clauses on your own.) The equijoin (ae.usubjid=ex.usubjid) reduces that to 84 records (as shown in Appendix 1.) Adding the restriction that the event has to occur before a dosing to the WHERE clause removes another 35 records. The GROUP BY and HAVING clauses operate on those 49 records, and keep the latest exposure datetime for each adverse event for each subject, giving us the final 9 records in Table 3.

This recipe will always work if you start with the proper ingredients, and follow the steps outlined in this paper.

RECOMMENDED READING

- *PROC SQL: Beyond the Basics Using SAS®, Third Edition*
- *Base SAS® Procedures Guide*

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Derek Morgan
mrdatesandtimes@gmail.com

APPENDICES: INTERMEDIATE DATASETS

Appendix 1: Complete Equijoin

usubjid	aeseq	exseq	exstdt	lastdose
0001	**1**	1	01DEC2018	17
0001	**2**	1	01DEC2018	12
0001	**3**	1	01DEC2018	8
0001	**4**	1	01DEC2018	38
0001	**5**	1	01DEC2018	70
0001	**6**	1	01DEC2018	41
0001	**1**	2	08DEC2018	10
0001	**2**	2	08DEC2018	5
0001	**3**	2	08DEC2018	1
0001	**4**	2	08DEC2018	31
0001	**5**	2	08DEC2018	63
0001	**6**	2	08DEC2018	34
0001	**1**	3	15DEC2018	3
0001	**2**	3	15DEC2018	-2
0001	**3**	3	15DEC2018	-6
0001	**4**	3	15DEC2018	24
0001	**5**	3	15DEC2018	56
0001	**6**	3	15DEC2018	27
0001	**1**	4	22DEC2018	-4
0001	**2**	4	22DEC2018	-9
0001	**3**	4	22DEC2018	-13
0001	**4**	4	22DEC2018	17
0001	**5**	4	22DEC2018	49
0001	**6**	4	22DEC2018	20
0001	**1**	5	29DEC2018	-11
0001	**2**	5	29DEC2018	-16
0001	**3**	5	29DEC2018	-20
0001	**4**	5	29DEC2018	10
0001	**5**	5	29DEC2018	42
0001	**6**	5	29DEC2018	13
0001	**1**	6	12JAN2019	-25
0001	**2**	6	12JAN2019	-30
0001	**3**	6	12JAN2019	-34
0001	**4**	6	12JAN2019	-4

usubjid	aeseq	exseq	exstdt	lastdose
0001	**5**	6	12JAN2019	28
0001	**6**	6	12JAN2019	-1
0001	**1**	7	19JAN2019	-32
0001	**2**	7	19JAN2019	-37
0001	**3**	7	19JAN2019	-41
0001	**4**	7	19JAN2019	-11
0001	**5**	7	19JAN2019	21
0001	**6**	7	19JAN2019	-8
0001	**1**	8	26JAN2019	-39
0001	**2**	8	26JAN2019	-44
0001	**3**	8	26JAN2019	-48
0001	**4**	8	26JAN2019	-18
0001	**5**	8	26JAN2019	14
0001	**6**	8	26JAN2019	-15
0001	**1**	9	02FEB2019	-46
0001	**2**	9	02FEB2019	-51
0001	**3**	9	02FEB2019	-55
0001	**4**	9	02FEB2019	-25
0001	**5**	9	02FEB2019	7
0001	**6**	9	02FEB2019	-22
0002	**1**	1	21MAR2019	48
0002	**2**	1	21MAR2019	58
0002	**3**	1	21MAR2019	45
0002	**1**	2	28MAR2019	41
0002	**2**	2	28MAR2019	51
0002	**3**	2	28MAR2019	38
0002	**1**	3	04APR2019	34
0002	**2**	3	04APR2019	44
0002	**3**	3	04APR2019	31
0002	**1**	4	11APR2019	27
0002	**2**	4	11APR2019	37
0002	**3**	4	11APR2019	24
0002	**1**	5	18APR2019	20
0002	**2**	5	18APR2019	30
0002	**3**	5	18APR2019	17
0002	**1**	6	25APR2019	13

usubjid	aeseq	exseq	exstdt	lastdose
0002	**2**	6	25APR2019	23
0002	**3**	6	25APR2019	10
0002	**1**	7	02MAY2019	6
0002	**2**	7	02MAY2019	16
0002	**3**	7	02MAY2019	3
0002	**1**	8	09MAY2019	-1
0002	**2**	8	09MAY2019	9
0002	**3**	8	09MAY2019	-4
0002	**1**	9	16MAY2019	-8
0002	**2**	9	16MAY2019	2
0002	**3**	9	16MAY2019	-11
0002	**1**	10	23MAY2019	-15
0002	**2**	10	23MAY2019	-5
0002	**3**	10	23MAY2019	-18

Appendix 2: Enhanced Equijoin with Date Filter Added

usubjid	aeseq	exseq	astdt	astdtm	exstdt	exstdtm	lastdose
0001	**1**	1	18DEC2018	.	01DEC2018	01DEC2018:00:07:00	17
0001	**2**	1	13DEC2018	.	01DEC2018	01DEC2018:00:07:00	12
0001	**3**	1	09DEC2018	.	01DEC2018	01DEC2018:00:07:00	8
0001	**4**	1	08JAN2019	.	01DEC2018	01DEC2018:00:07:00	38
0001	**5**	1	09FEB2019	09FEB2019:05:54:00	01DEC2018	01DEC2018:00:07:00	70
0001	**6**	1	11JAN2019	11JAN2019:11:35:00	01DEC2018	01DEC2018:00:07:00	41
0001	**1**	2	18DEC2018	.	08DEC2018	08DEC2018:00:02:00	10
0001	**2**	2	13DEC2018	.	08DEC2018	08DEC2018:00:02:00	5
0001	**3**	2	09DEC2018	.	08DEC2018	08DEC2018:00:02:00	1
0001	**4**	2	08JAN2019	.	08DEC2018	08DEC2018:00:02:00	31
0001	**5**	2	09FEB2019	09FEB2019:05:54:00	08DEC2018	08DEC2018:00:02:00	63
0001	**6**	2	11JAN2019	11JAN2019:11:35:00	08DEC2018	08DEC2018:00:02:00	34
0001	**1**	3	18DEC2018	.	15DEC2018	15DEC2018:00:06:00	3
0001	**4**	3	08JAN2019	.	15DEC2018	15DEC2018:00:06:00	24
0001	**5**	3	09FEB2019	09FEB2019:05:54:00	15DEC2018	15DEC2018:00:06:00	56
0001	**6**	3	11JAN2019	11JAN2019:11:35:00	15DEC2018	15DEC2018:00:06:00	27
0001	**4**	4	08JAN2019	.	22DEC2018	22DEC2018:00:10:00	17
0001	**5**	4	09FEB2019	09FEB2019:05:54:00	22DEC2018	22DEC2018:00:10:00	49
0001	**6**	4	11JAN2019	11JAN2019:11:35:00	22DEC2018	22DEC2018:00:10:00	20
0001	**4**	5	08JAN2019	.	29DEC2018	29DEC2018:00:09:00	10
0001	**5**	5	09FEB2019	09FEB2019:05:54:00	29DEC2018	29DEC2018:00:09:00	42
0001	**6**	5	11JAN2019	11JAN2019:11:35:00	29DEC2018	29DEC2018:00:09:00	13
0001	**5**	6	09FEB2019	09FEB2019:05:54:00	12JAN2019	12JAN2019:00:01:00	28
0001	**5**	7	09FEB2019	09FEB2019:05:54:00	19JAN2019	19JAN2019:00:04:00	21
0001	**5**	8	09FEB2019	09FEB2019:05:54:00	26JAN2019	26JAN2019:00:07:00	14
0001	**5**	9	09FEB2019	09FEB2019:05:54:00	02FEB2019	02FEB2019:00:05:00	7
0002	**1**	1	08MAY2019	08MAY2019:08:30:00	21MAR2019	21MAR2019:00:06:00	48
0002	**2**	1	18MAY2019	18MAY2019:09:28:00	21MAR2019	21MAR2019:00:06:00	58
0002	**3**	1	05MAY2019	05MAY2019:03:19:00	21MAR2019	21MAR2019:00:06:00	45
0002	**1**	2	08MAY2019	08MAY2019:08:30:00	28MAR2019	28MAR2019:00:04:00	41
0002	**2**	2	18MAY2019	18MAY2019:09:28:00	28MAR2019	28MAR2019:00:04:00	51
0002	**3**	2	05MAY2019	05MAY2019:03:19:00	28MAR2019	28MAR2019:00:04:00	38
0002	**1**	3	08MAY2019	08MAY2019:08:30:00	04APR2019	04APR2019:00:05:00	34
0002	**2**	3	18MAY2019	18MAY2019:09:28:00	04APR2019	04APR2019:00:05:00	44
0002	**3**	3	05MAY2019	05MAY2019:03:19:00	04APR2019	04APR2019:00:05:00	31
0002	**1**	4	08MAY2019	08MAY2019:08:30:00	11APR2019	11APR2019:00:10:00	27

usubjid	aeseq	exseq	astdt	astdtm	exstdt	exstdtm	lastdose
0002	**2**	4	18MAY2019	18MAY2019:09:28:00	11APR2019	11APR2019:00:10:00	37
0002	**3**	4	05MAY2019	05MAY2019:03:19:00	11APR2019	11APR2019:00:10:00	24
0002	**1**	5	08MAY2019	08MAY2019:08:30:00	18APR2019	18APR2019:00:02:00	20
0002	**2**	5	18MAY2019	18MAY2019:09:28:00	18APR2019	18APR2019:00:02:00	30
0002	**3**	5	05MAY2019	05MAY2019:03:19:00	18APR2019	18APR2019:00:02:00	17
0002	**1**	6	08MAY2019	08MAY2019:08:30:00	25APR2019	25APR2019:00:01:00	13
0002	**2**	6	18MAY2019	18MAY2019:09:28:00	25APR2019	25APR2019:00:01:00	23
0002	**3**	6	05MAY2019	05MAY2019:03:19:00	25APR2019	25APR2019:00:01:00	10
0002	**1**	7	08MAY2019	08MAY2019:08:30:00	02MAY2019	02MAY2019:00:07:00	6
0002	**2**	7	18MAY2019	18MAY2019:09:28:00	02MAY2019	02MAY2019:00:07:00	16
0002	**3**	7	05MAY2019	05MAY2019:03:19:00	02MAY2019	02MAY2019:00:07:00	3
0002	**2**	8	18MAY2019	18MAY2019:09:28:00	09MAY2019	09MAY2019:00:02:00	9
0002	**2**	9	18MAY2019	18MAY2019:09:28:00	16MAY2019	16MAY2019:00:01:00	2