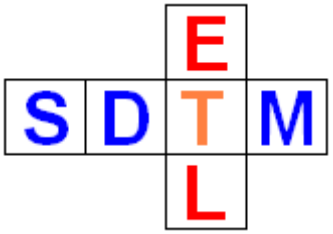
SDTM-ETL 4.x/5.x: Different ways for filtering:
adding XPath conditions, relative XPath-s,
and using the "xpathfilter" feature

Author: Jozef Aerts, XML4Pharma



Last update: **2026-02-12**

Table of Contents

Introduction

In most cases, using drag-and-drop and following the wizards, including in-/excluding visits (StudyEvent), forms, subforms and items will give the user everything that is needed, and nothing needs to be changed in the XPath expressions that extract the data from the ODM to populate SDTM/SEND variables.

There are however some very special cases where users want to change or further refine the XPath expressions to retrieve the data, especially for the "looping variable" (which usually is the SDTM/SEND "topic variable"). In such cases, the use of relative XPaths for the other variables can be helpful.

In this tutorial we will explain the different possibilities.

Example case

One of such very special cases is when one has (relative) low-quality data, e.g. data that should not have made it into the ODM anyway. Of course the best way in such cases is to take care that a "clean" database is used for the ODM generation. This is however not always possible, e.g. when the database is not cleaned and closed yet, and data is still being added. Essentially, this is not bad at all, as it allows to have temporary SDTM/SEND data even while the study is still running.

Just for the example (don't pin us on whether this is a real scenario ...), we will use the case that some of the erythrocytes and leukocytes data points in the ODM file with the clinical data have the value "UNKNOWN", e.g.

```
<FormData FormOID="F_LAB">
    <ItemGroupData ItemGroupOID="IG_COMMON">
        <ItemData ItemOID="I_SITE" Value="23"/>
        <ItemData ItemOID="I_SUBJECTID" Value="001"/>
        <ItemData ItemOID="I_VISIT" Value="2010-02-27"/>
        <ItemData ItemOID="I_VISITTIME" Value="10:49:23"/>
    </ItemGroupData>
    <ItemGroupData ItemGroupOID="IG_LB_HEMATOLOGY">
        <ItemData ItemOID="I_LB_NAME" Value="Singen Central Lab"/>
        <ItemData ItemOID="I_LB_ID" Value="SIN"/>
        <ItemData ItemOID="I_LB_ACCESSION" Value="3076"/>
        <ItemData ItemOID="I_LB_RBC_NOTDONE" Value="false"/>
        <ItemData ItemOID="I_LB_RBC" Value="UNKNOWN"/>
        <ItemData ItemOID="I_LB_RBC_LO" Value="4.7"/>
        <ItemData ItemOID="I_LB_RBC_HI" Value="6.1"/>
        <ItemData ItemOID="I_LB_WBC_NOTDONE" Value="false"/>
        <ItemData ItemOID="I_LB_WBC" Value="6.2"/>
        <ItemData ItemOID="I_LB_WBC_LO" Value="4.3"/>
        <ItemData ItemOID="I_LB_WBC_HI" Value="10.8"/>
    </ItemGroupData>
```

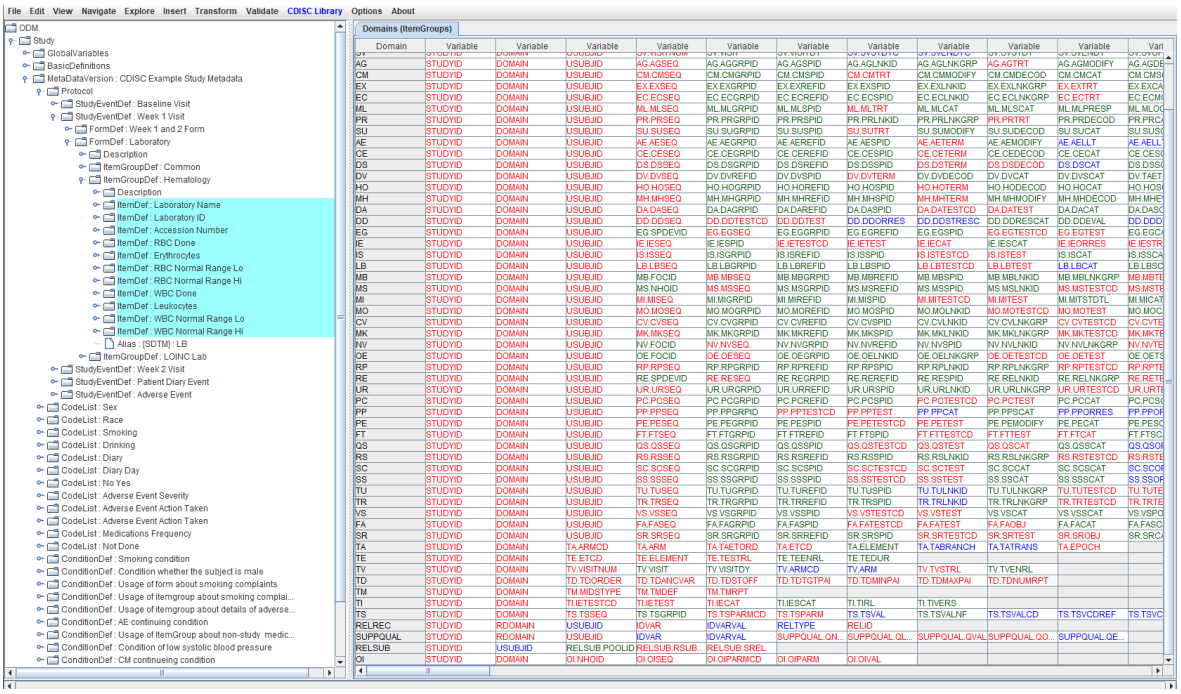
and we want to exclude these from the generated SDTM.

In a second case, we will show how to exclude a subject without changing anything in the ODM file with clinical data.

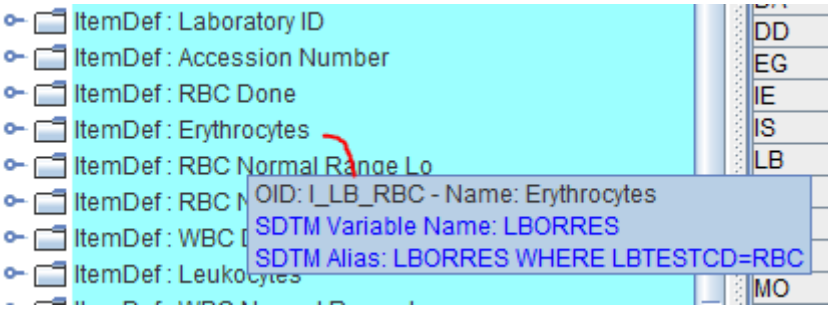
Making simple selections through the GUI dialogs

Excluding specific data points by their value

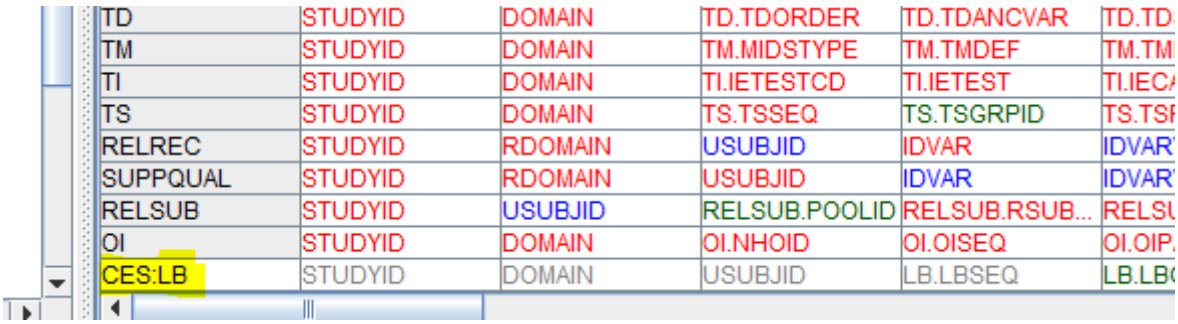
When having loaded the ODM metadata and having loaded an SDTM (or SEND) template, we find the following in the main window:



the ODM items on the left having a cyan background as they have SDTM annotations about where they belong in SDTM, which is of course great. For example:

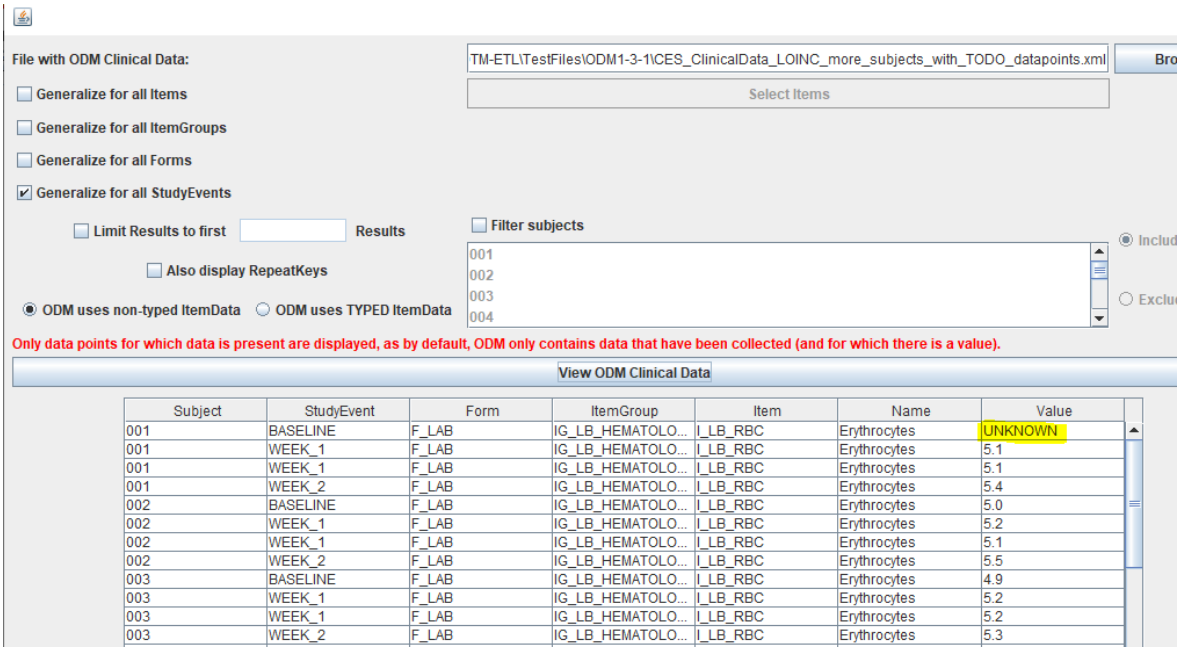


On the right side, the SDTM table, we can now drag-and-drop the "LB" row from the template to the bottom (or use the menu "Edit - Copy domain/dataset" and "Edit - Paste domain dataset"), leading to:

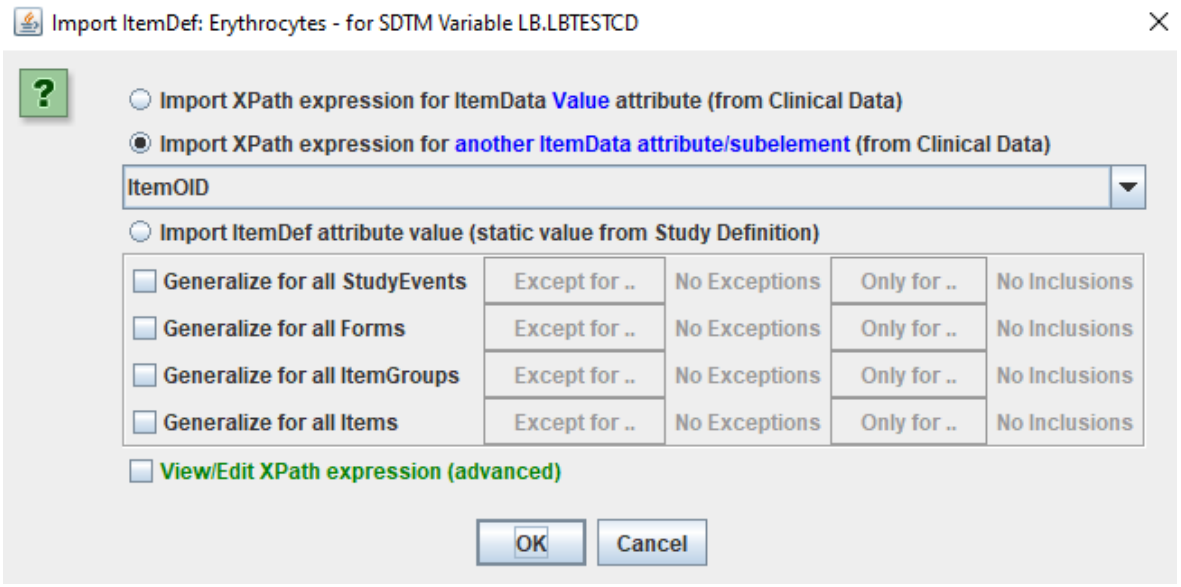


We call this a "study-specific instance of LB".

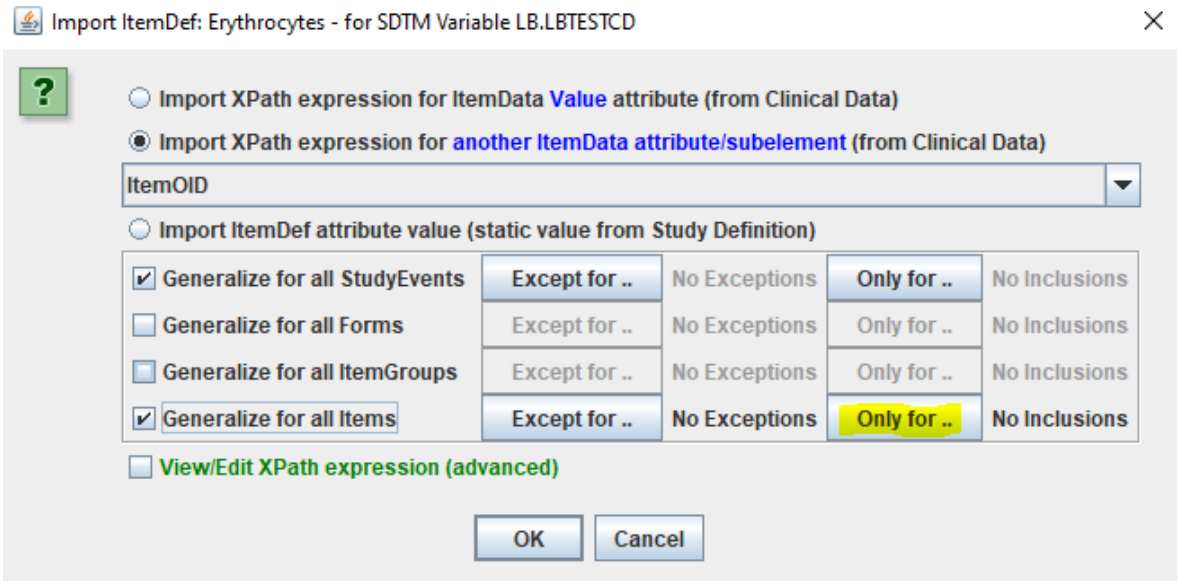
Now, if we select "Erythrocytes" on the left in the ODM tree, and select our ODM file with the clinical data, and then use the menu "View - ODM Clinical Data" (with "Generalize for all StudyEvents" checked, we find:



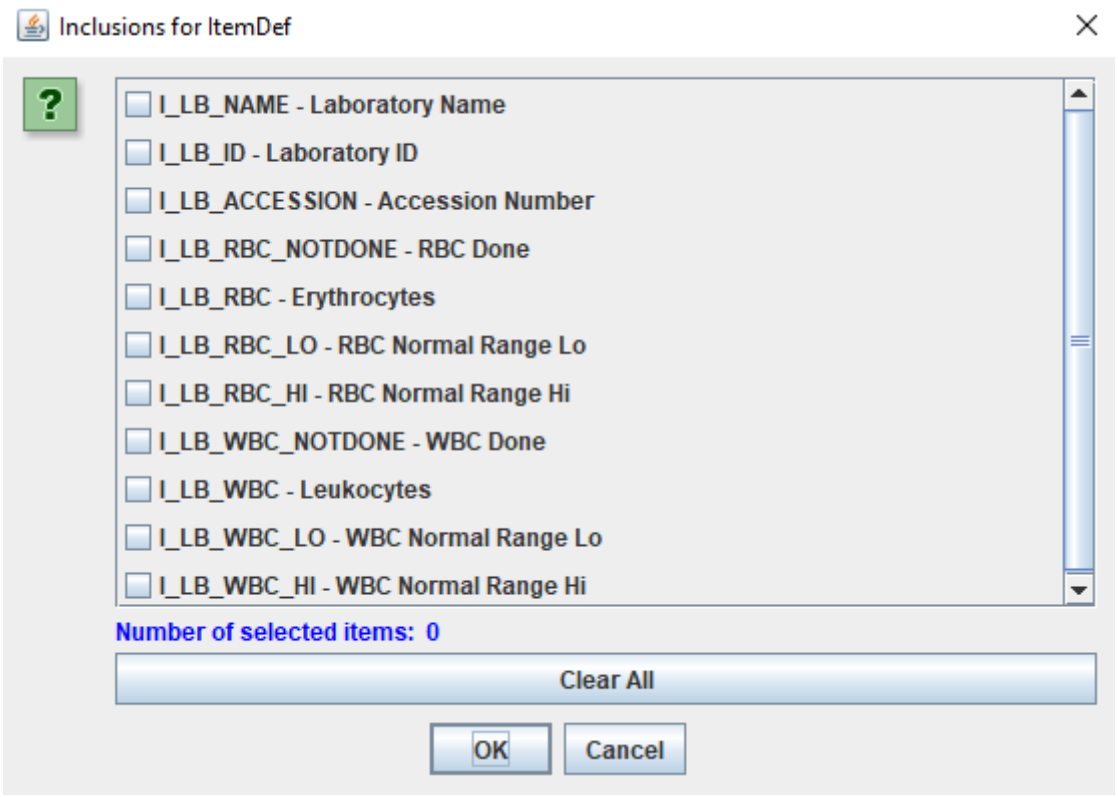
As always, we start with generating the mappings for the SDTM "topic" variable, which is in this case LBTESTCD. So we drag-and-drop from "Erythrocytes" in the ODM tree on the lefz to the SDTM cell "LBTESTCD" on the bottom right. This then leads to:



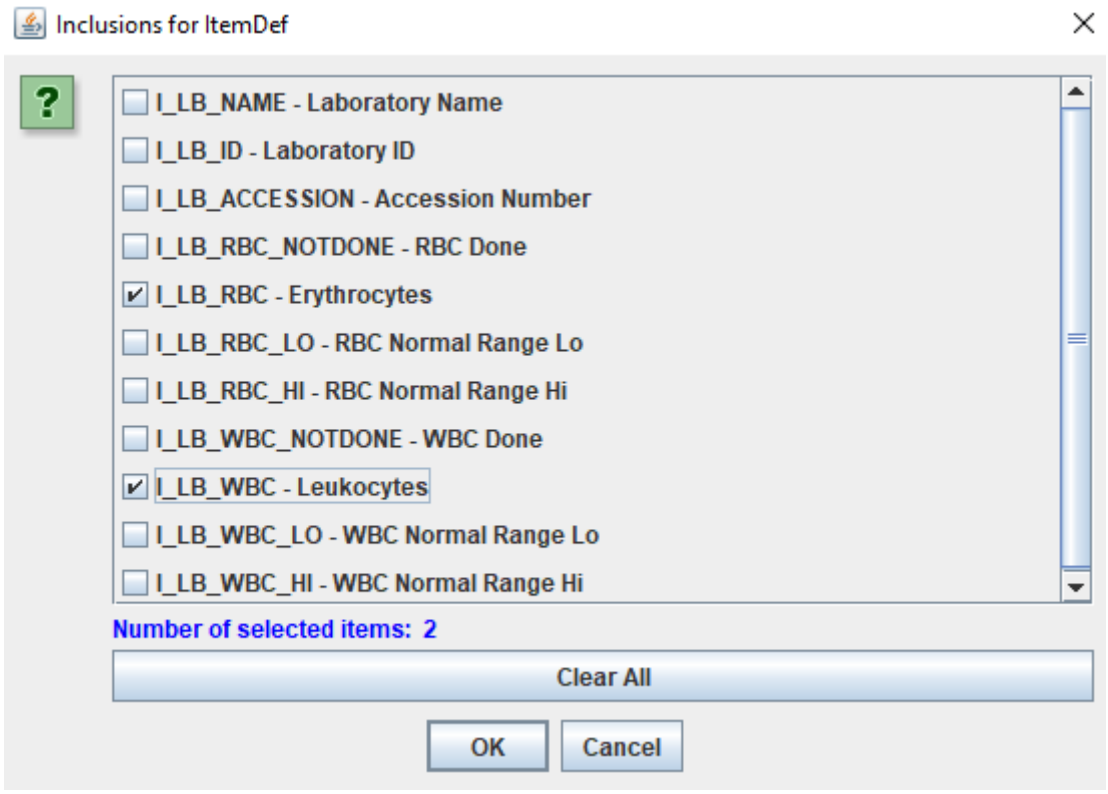
As the system knows that we do not want to get the collected value (which is a number), but the identifier of the item, it already suggests "Import XPath …" for "ItemOID".
As we want to get the items for all visits (named "StudyEvents" in ODM), we check the checkbox "Generalize for all StudyEvents". We also want to also include "Leukocytes", so we check "Generalize for all Items" and then click "Only for …":



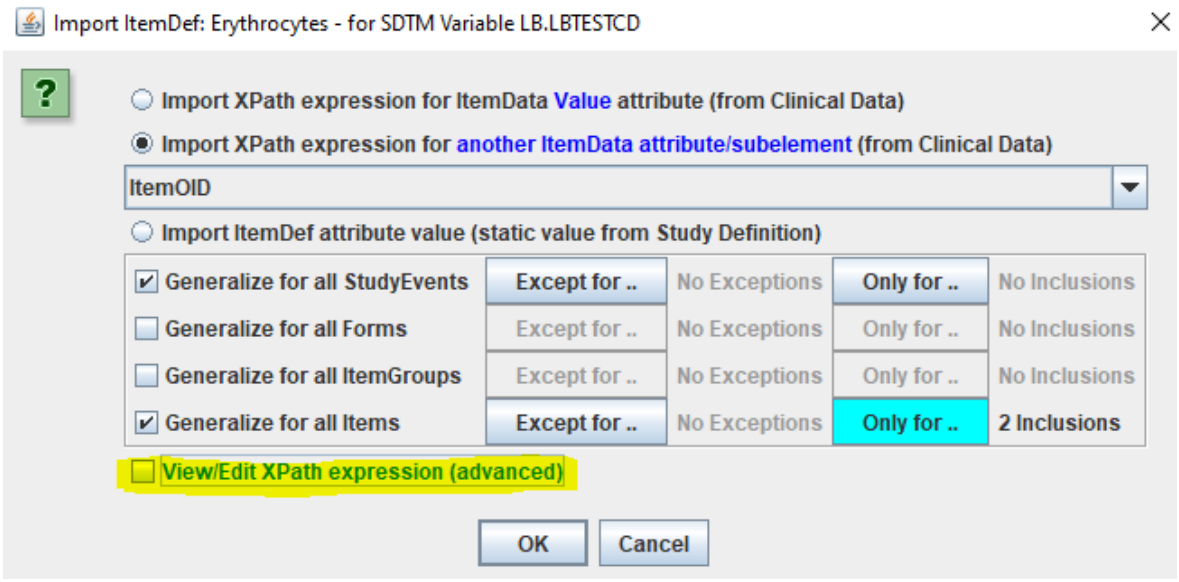
leading to:



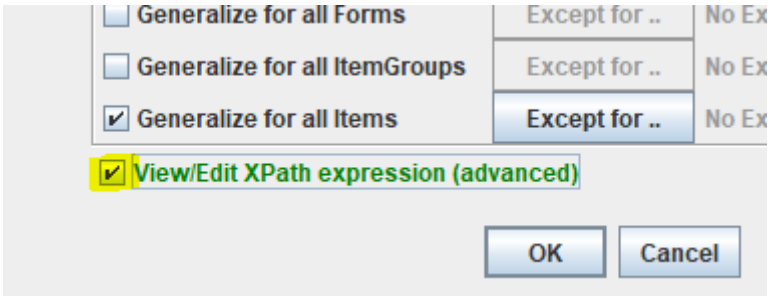
we check "Erythrocytes" and "Leukocytes" to take care that only these are selected, as we do not want to e.g. have "Laboratory ID" mapped to LBTESTCD:



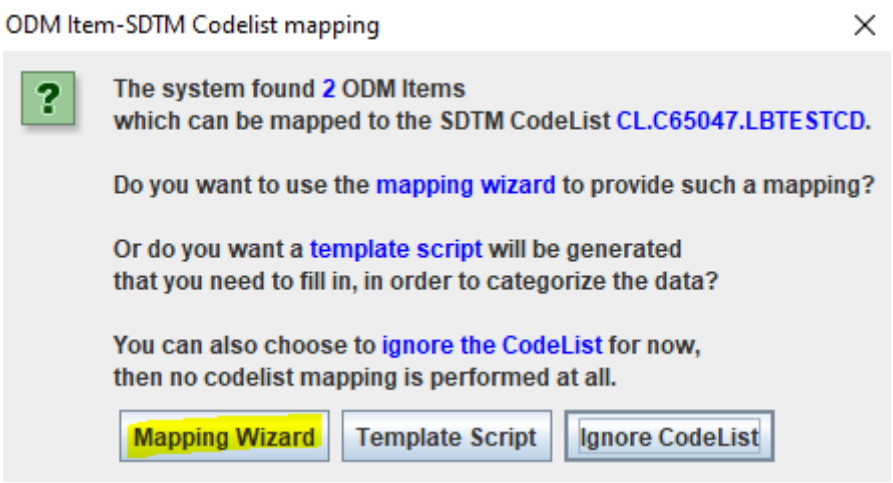
After clicking "OK", we get:



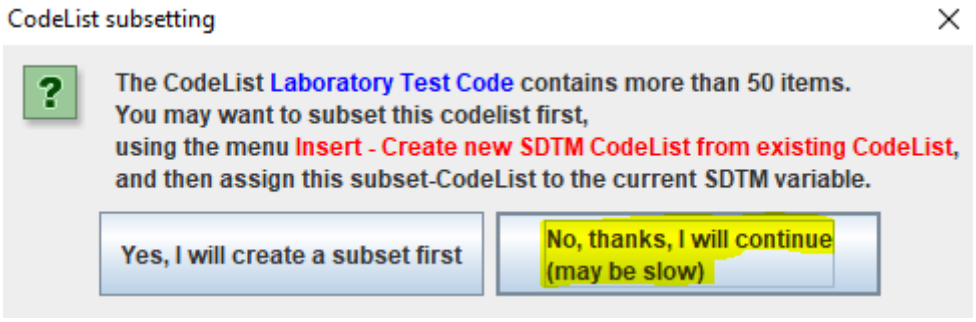
Already notice the checkbox "View/Edit XPath expression …". If we check it:



and then click "OK", we get:

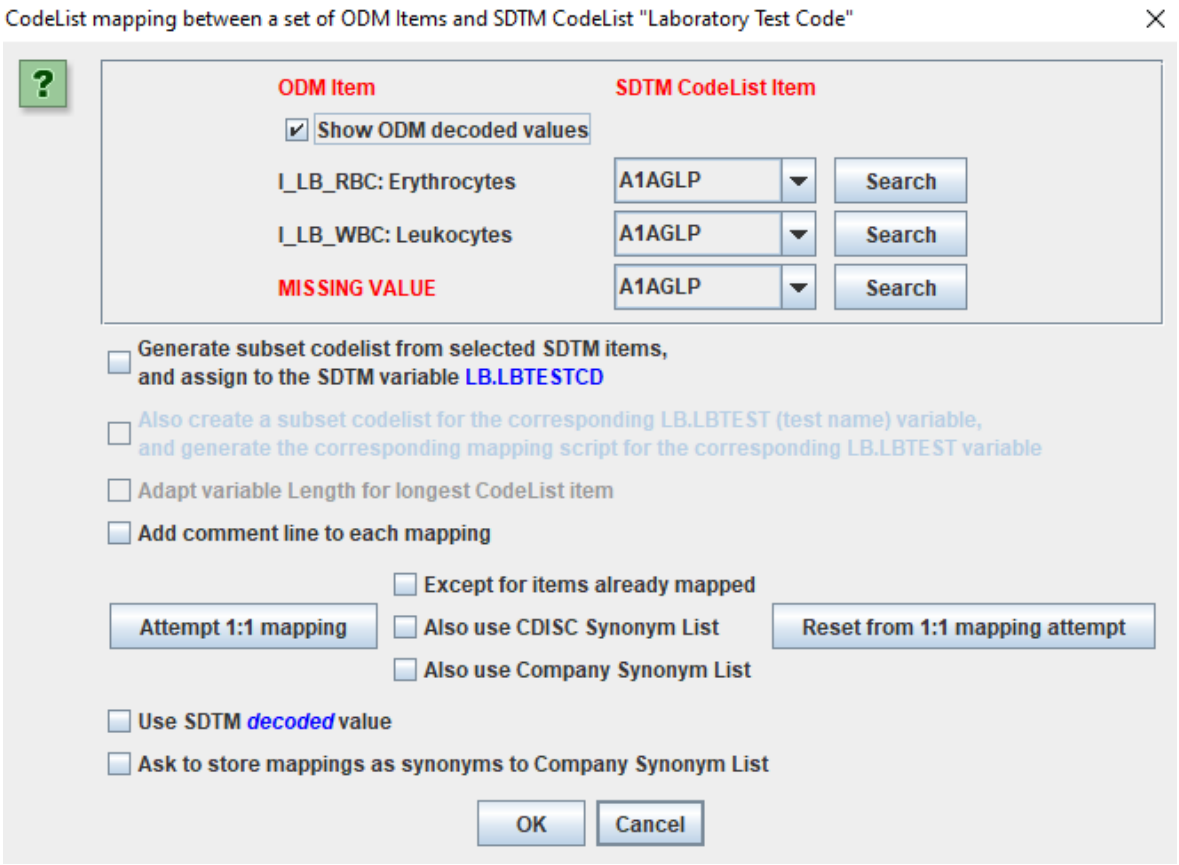


As very often, we will use the "Mapping Wizard", so we click that button, leading to:

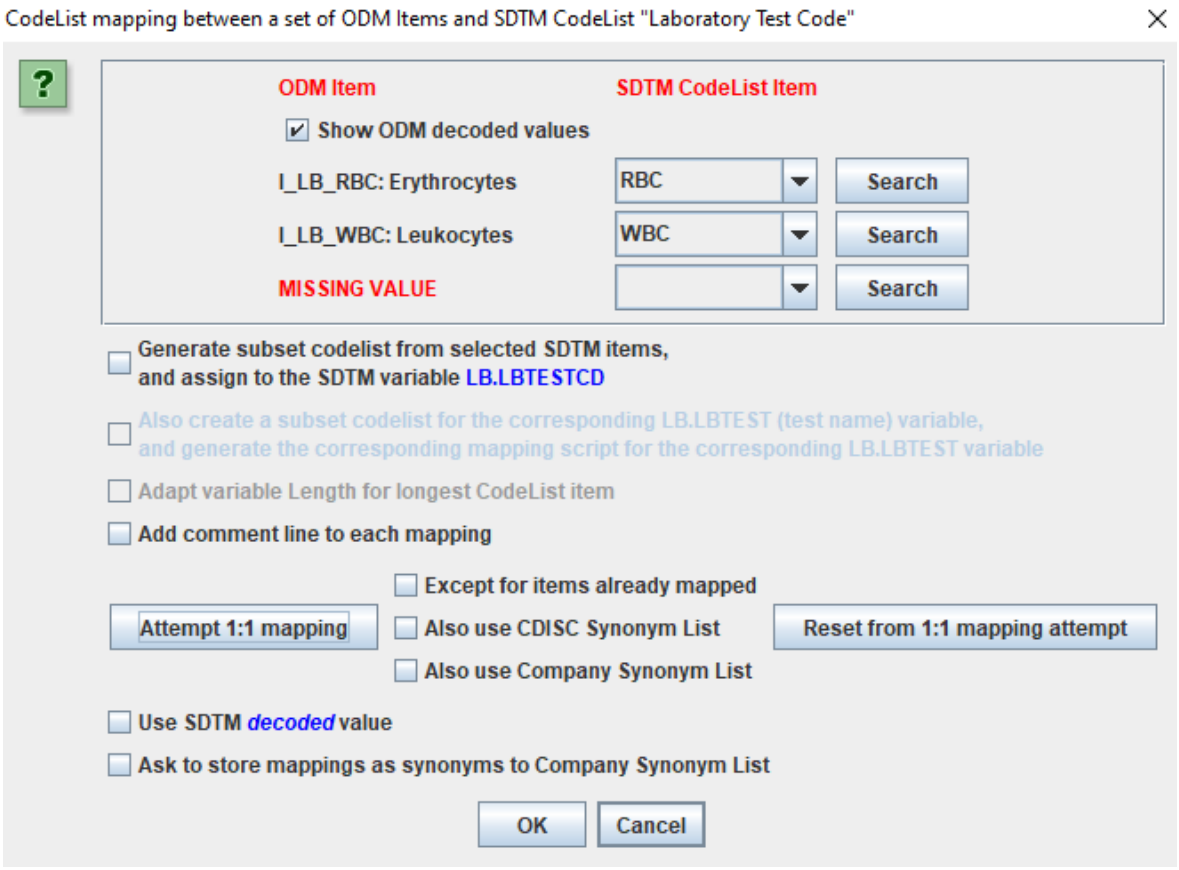


As the SDTM associated codelist is large, it suggests us to first generate a subset codelist, but we are lazy, and just click "No, thanks …", leading to the next "wizard"[1]:

[1] As the SDTM codelist for LBTESTCD is very large, this may take a little bit of time.

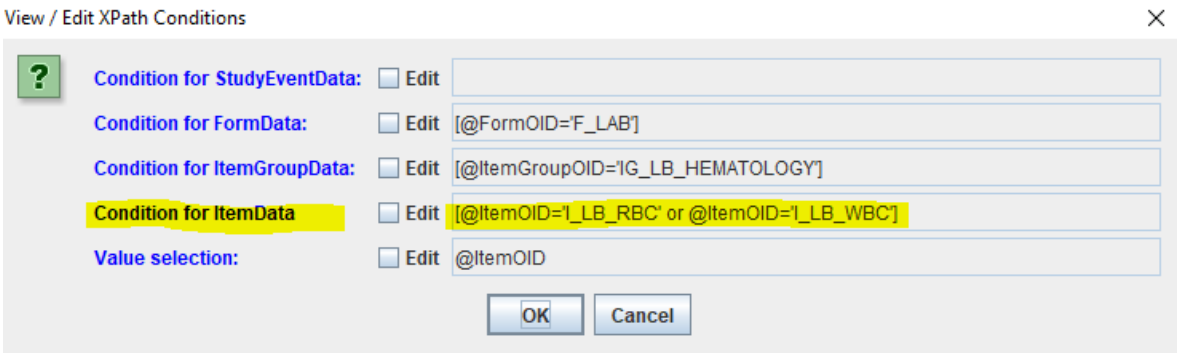


with the checkbox "Show ODM decoded values" allowing us to see some more information for the case we want to assign values for LBTESTCD using the dropboxes on the right.
We are however once again lazy and just use the button "Attempt 1:1 mapping", leading to:



which looks very good[2].

and when we then click "OK", as we checked "View/Edit XPath expression ..." before, we get:



Notice the "Condition" for ItemData", which we will use later.

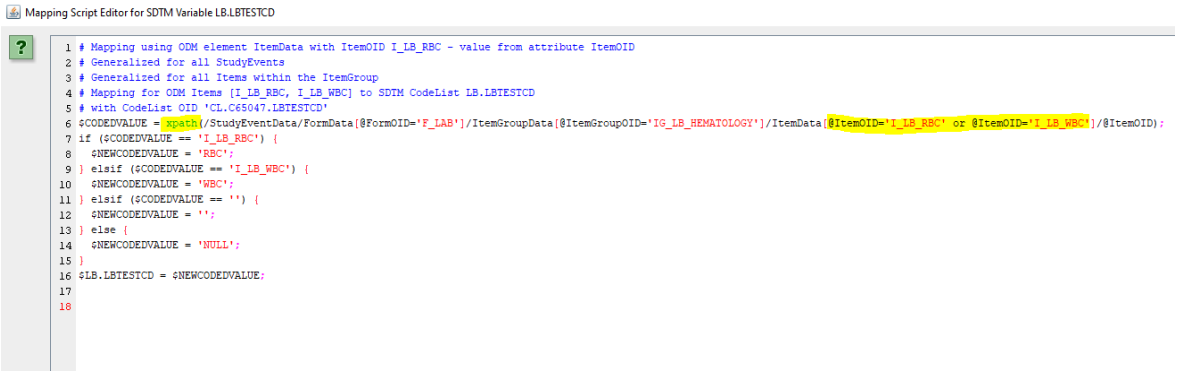
Clicking "OK" then leads to the generation of the mapping script:

[2] The "Attempt 1:1 mapping" button starts an algorithm based on word similarity.

The Transformation Script

```
# Mapping using ODM element ItemData with ItemOID I_LB_RBC - value from attribute ItemOID
# Generalized for all StudyEvents
# Generalized for all Items within the ItemGroup
# Mapping for ODM Items [I_LB_RBC, I_LB_WBC] to SDTM CodeList LB.LBTESTCD
# with CodeList OID 'CL.C65047.LBTESTCD'
$CODEDVALUE = xpath(/StudyEventData/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGroupOID='IG_LB_HEMATOLOGY']/
if ($CODEDVALUE == 'I_LB_RBC') {
  $NEWCODEDVALUE = 'RBC';
} elsif ($CODEDVALUE == 'I_LB_WBC') {
  $NEWCODEDVALUE = 'WBC';
} elsif ($CODEDVALUE == '') {
  $NEWCODEDVALUE = '';
} else {
  $NEWCODEDVALUE = 'NULL';
}
$LB.LBTESTCD = $NEWCODEDVALUE;
```

and if we use the "Full-screen" button:



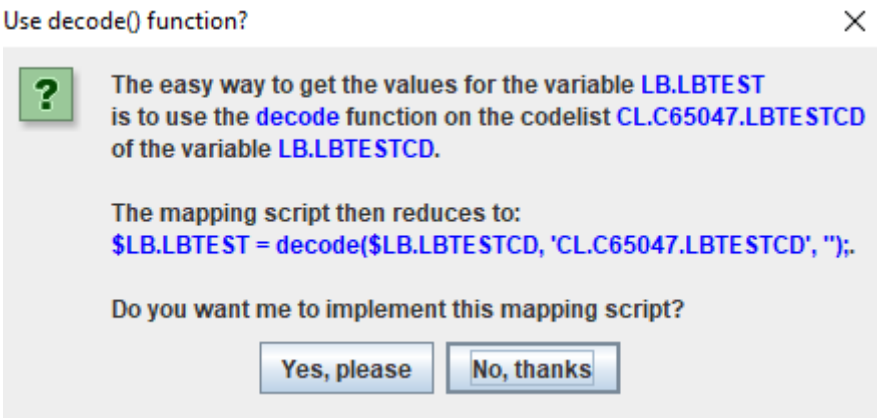
nicely showing that the XPath expression will only select items for "Erythrocytes" (RBC) and "Leukocytes".

We can now continue with adding the mappings for other major variables such as "LBTEST" and "LBORRES".

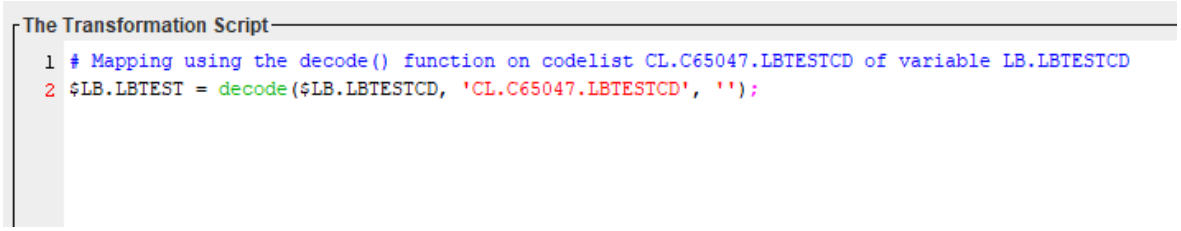
After clicking "OK" several times to return the main window, we find:

TR	STUDYID	DOMAIN	USUBJID	TR.TRSEQ	TR.TRGRPID	TR.TRREFID	TR.TRSPID	TR.TRLNKID	TR.TRLNKGRP	TR
VS	STUDYID	DOMAIN	USUBJID	VS.VSSEQ	VS.VSGRPID	VS.VSSPID	VS.VSTESTCD	VS.VSTEST	VS.VSCAT	VS
FA	STUDYID	DOMAIN	USUBJID	FA.FASEQ	FA.FAGRPID	FA.FASPID	FA.FATESTCD	FA.FATEST	FA.FAOBJ	FA
SR	STUDYID	DOMAIN	USUBJID	SR.SRSEQ	SR.SRGRPID	SR.SRREFID	SR.SRSPID	SR.SRTESTCD	SR.SRTEST	SR
TA	STUDYID	DOMAIN	TA.ARMCD	TA.ARM	TA.TAETORD	TA.ETCD	TA.ELEMENT	TA.TABRANCH	TA.TATRANS	TA
TE	STUDYID	DOMAIN	TE.ETCD	TE.ELEMENT	TE.TESTRL	TE.TEENRL	TE.TEDUR			
TV	STUDYID	DOMAIN	TV.VISITNUM	TV.VISIT	TV.VISITDY	TV.ARMCD	TV.ARM	TV.TVSTRL	TV.TVENRL	
TD	STUDYID	DOMAIN	TD.TDORDER	TD.TDANCVAR	TD.TDSTOFF	TD.TDTGTPAI	TD.TDMINPAI	TD.TDMAXPAI	TD.TDNUMRPT	
TM	STUDYID	DOMAIN	TM.MIDSTYPE	TM.TMDEF	TM.TMRPT					
TI	STUDYID	DOMAIN	TI.IETESTCD	TI.IETEST	TI.IECAT	TI.IESCAT	TI.TIRL	TI.TIVERS		
TS	STUDYID	DOMAIN	TS.TSSEQ	TS.TSGRPID	TS.TSPARMCD	TS.TSPARM	TS.TSVAL	TS.TSVALNF	TS.TSVALCD	TS
RELREC	STUDYID	RDOMAIN	USUBJID	IDVAR	IDVARVAL	RELTYPE	RELID			
SUPPQUAL	STUDYID	RDOMAIN	USUBJID	IDVAR	IDVARVAL	SUPPQUAL.QN...	SUPPQUAL.QL...	SUPPQUAL.QVAL	SUPPQUAL.QO...	SU
RELSUB	STUDYID	USUBJID	RELSUB.POOLID	RELSUB.RSUB...	RELSUB.SREL					
OI	STUDYID	DOMAIN	OI.NHOID	OI.OISEQ	OI.OIPARMCD	OI.OIPARM	OI.OIVAL			
CES:LB	STUDYID	DOMAIN	USUBJID	LB.LBSEQ	LB.LBGRPID	LB.LBREFID	LB.LBSPID	LB.LBTESTCD	LB.LBTEST	LB

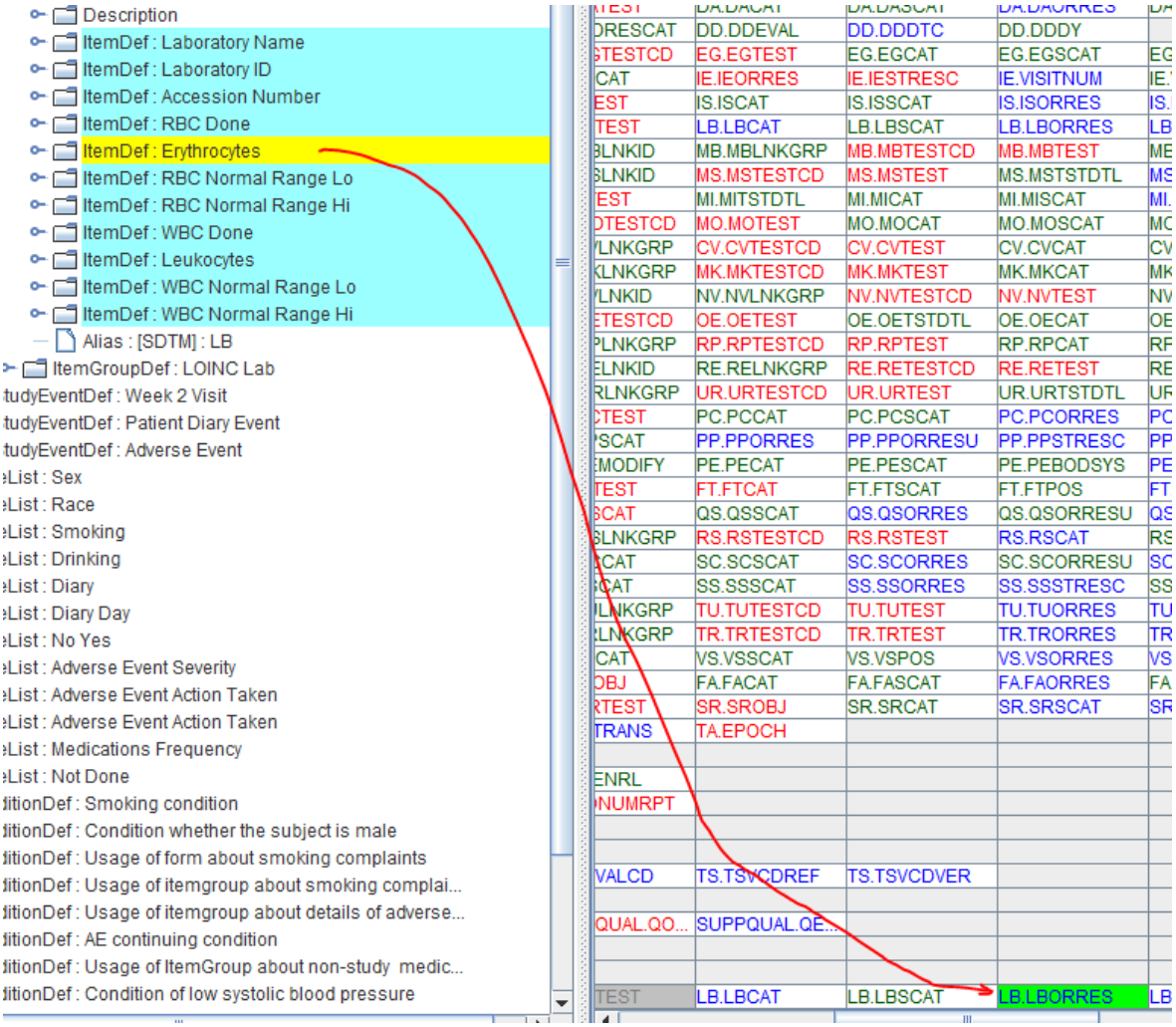
we see that the cell for LBTESTCD is "grayed out", meaning there is a mapping available, and the cell for LBTEST is still red, meaning it is a "required" SDTM variable. If we double click it, another wizard appears:



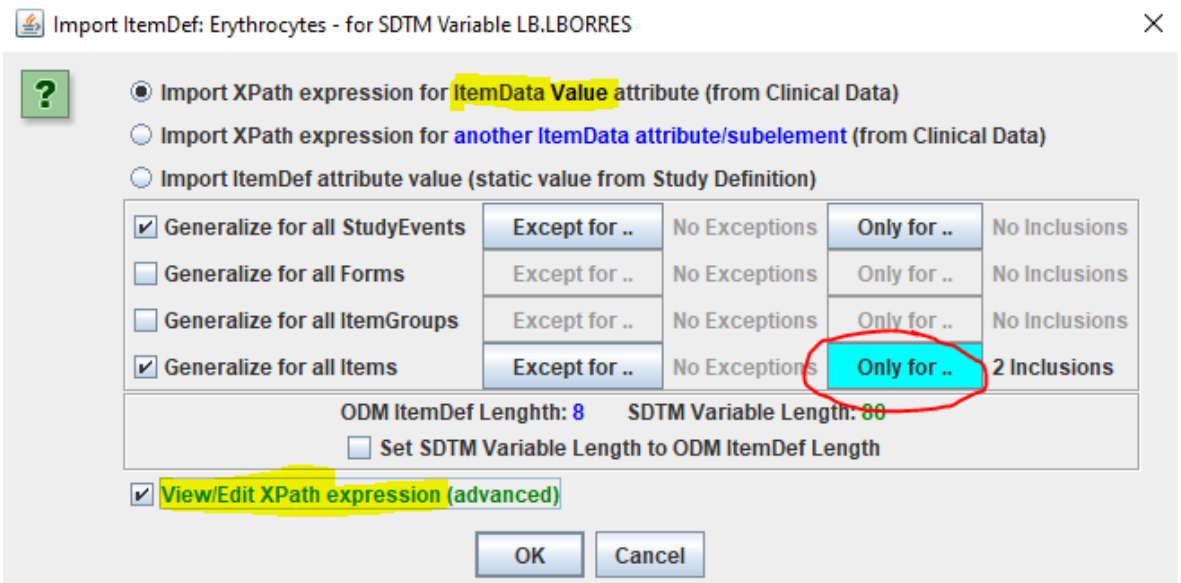
suggesting to just making it easy by using the "decode" value of LBTEST, and not to have to go through the whole process again. If we click "Yes, please", the mapping script is automatically created as:



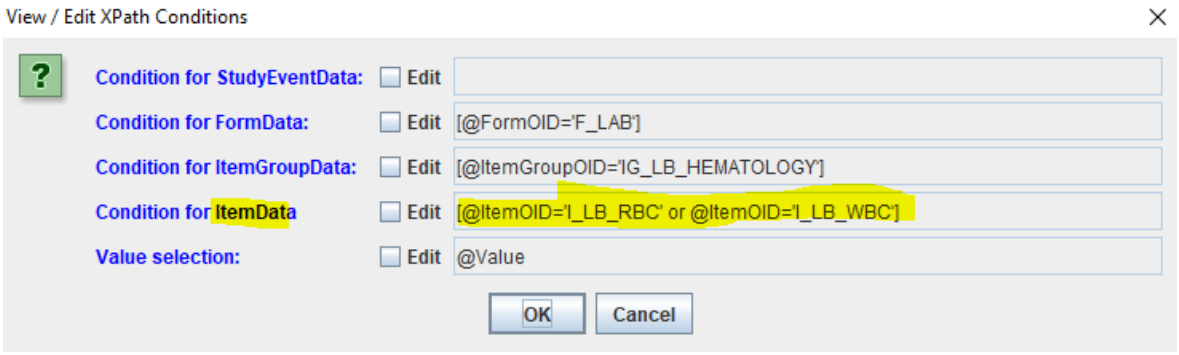
After clicking "OK", we can now drag-and-drop "Erythrocytes" from the ODM tree to the cell LBORRES:



and the first "wizard" is popping up again, but this time suggesting to take the "Value":



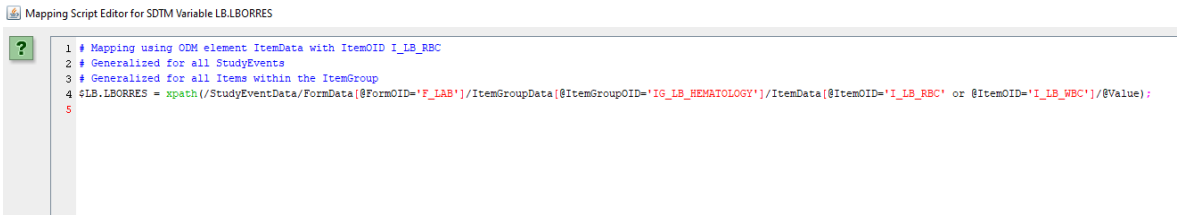
where we also see that the system has remembered that this should be applied to all visits (StudyEvents) and only for "Erythrocytes" and "Leukocytes".
We check the checkbox "View/Edit XPath expression" again, just for the demonstration for now. Clicking "OK" leads to:



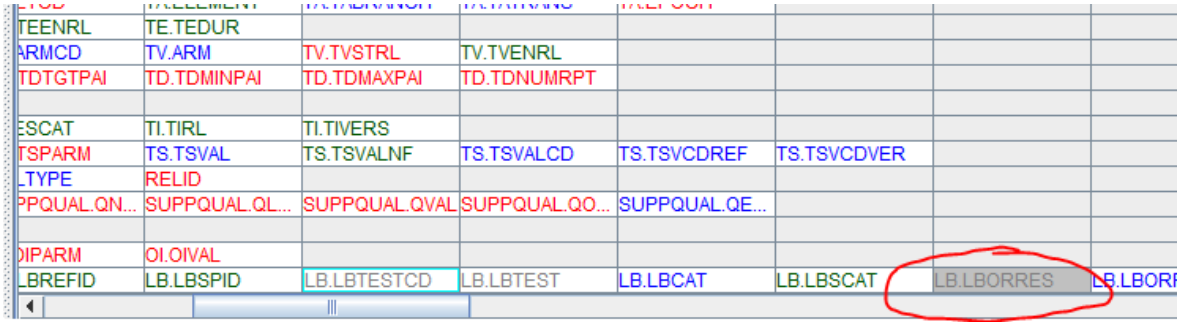
showing us the different "Selections" on each of the level of "StudyEvent" (visit), "Form", "ItemGroup" (subform) and "Item".

If we do not know XPath yet, we can here learn …:
everything that is between square brackets is a "condition" (in XPath they call it "predicate"), and thus can be compared e.g. with a "WHERE" in SQL.
So the line **[@ItemOID='I_LB_RBC' or @ItemOID='I_LB_WBC']**
means: select the items for which the identifier attribute (@ItemOID) is either "I_LB_RBC" or "I_LB_WBC", i.e. selecting the erythrocytes and leukocytes data points.

The clicking "OK" leads to the mapping script:

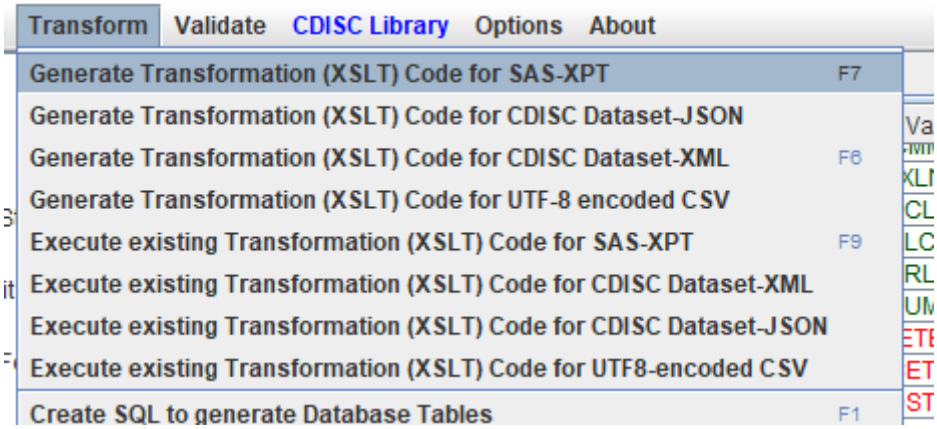


Clicking "OK" until we are in the main window, then we see that also "LBTEST" and "LBORRES" are "grayed out", as we have mappings for them now:

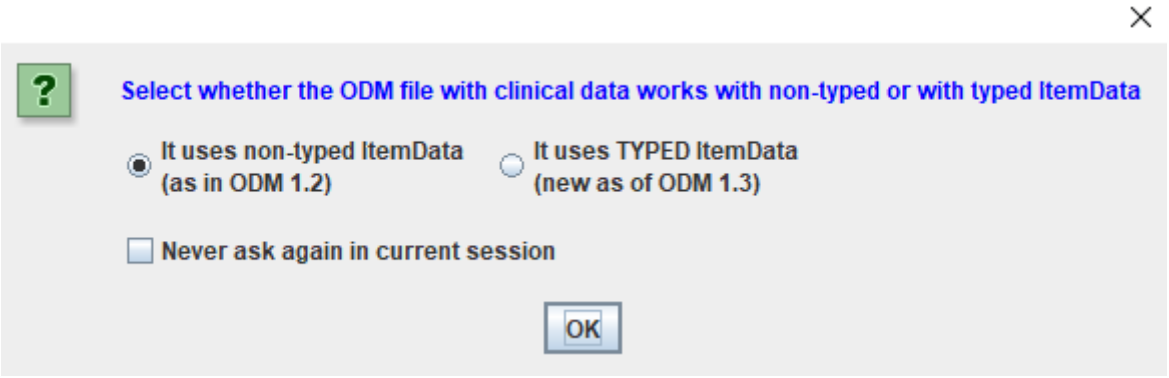


Fine! Let's generate some SDTM now …

For this, we use the menu "Transform - Generate Transformation (XSLT) Code …".
Depending on the version of SDTM-ETL, this is just one choice or divides between the different output formats. In 4.6 e.g. this still is:

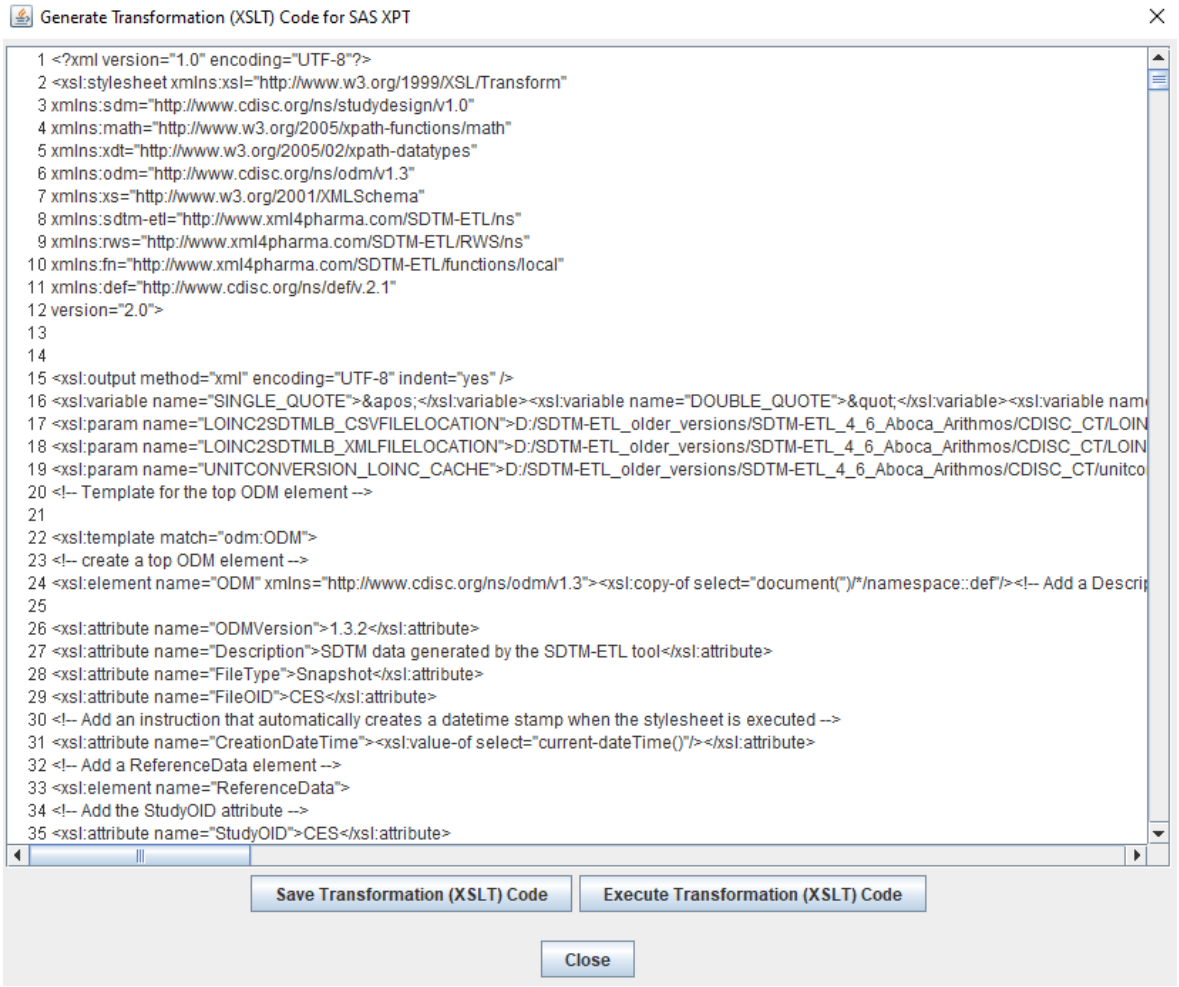


anyway leading to:



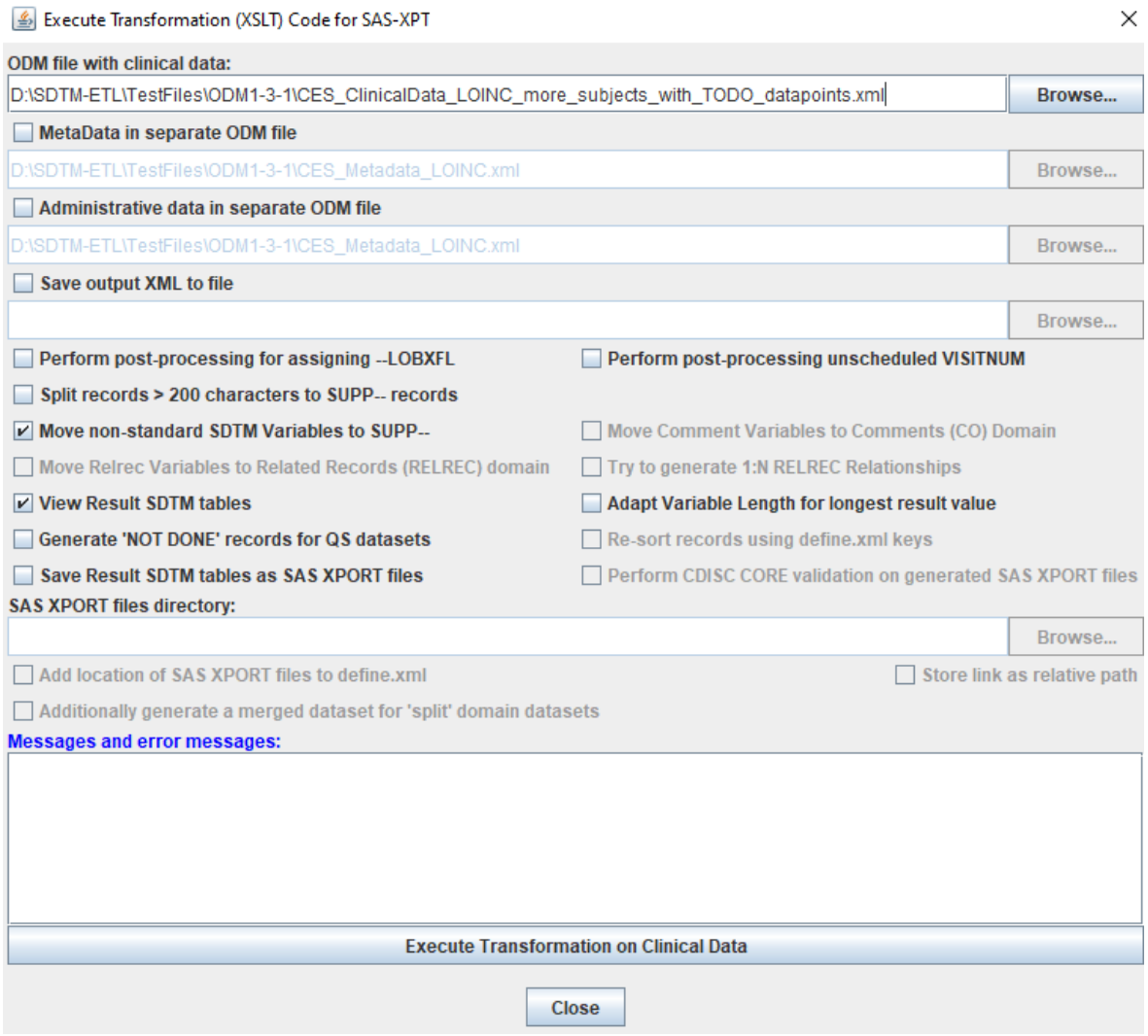
In 80% of the cases, you will have the "non-typed" ItemData as most of the EDC systems export this, whereas a minority of the EDC systems use "typed ItemData".
If you don't know, just ask them, or … just try …

This then leads to an intermediate screen:

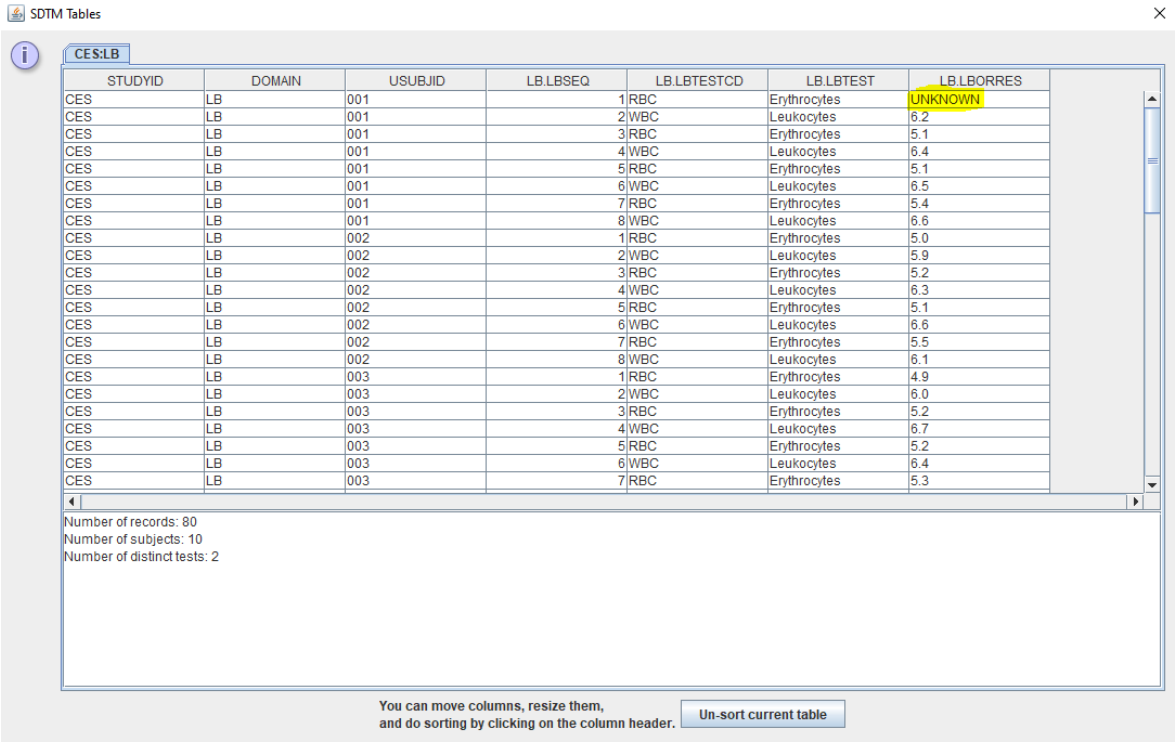


If you don't like it, you can skip this intermediate window using the menu "Options" and check the checkbox "Skip display of generated XSLT". Most users however don't mind (or really want to see the generated XSLT[3]). Click "Execute Transformation (XSLT) Code to continue, leading to:

[3] XSLT means "XML Stylesheet Transformation Language". See e.g.: https://www.w3schools.com/xml/xsl_intro.asp



Ensure that you have the correct file with clinical data in the filed "ODM file with clinical data" and then click "Execute Transformation on Clinical Data". This will then show the generated SDTM table:



There is no need yet to already generate SAS-XPT datasets (or Dataset-JSON datasets in bear future) as we are still "developing" …
We now see that we do have a data point with LBORRES is "UNKNOWN".
Also notice that the system states that 80 records have been generated.

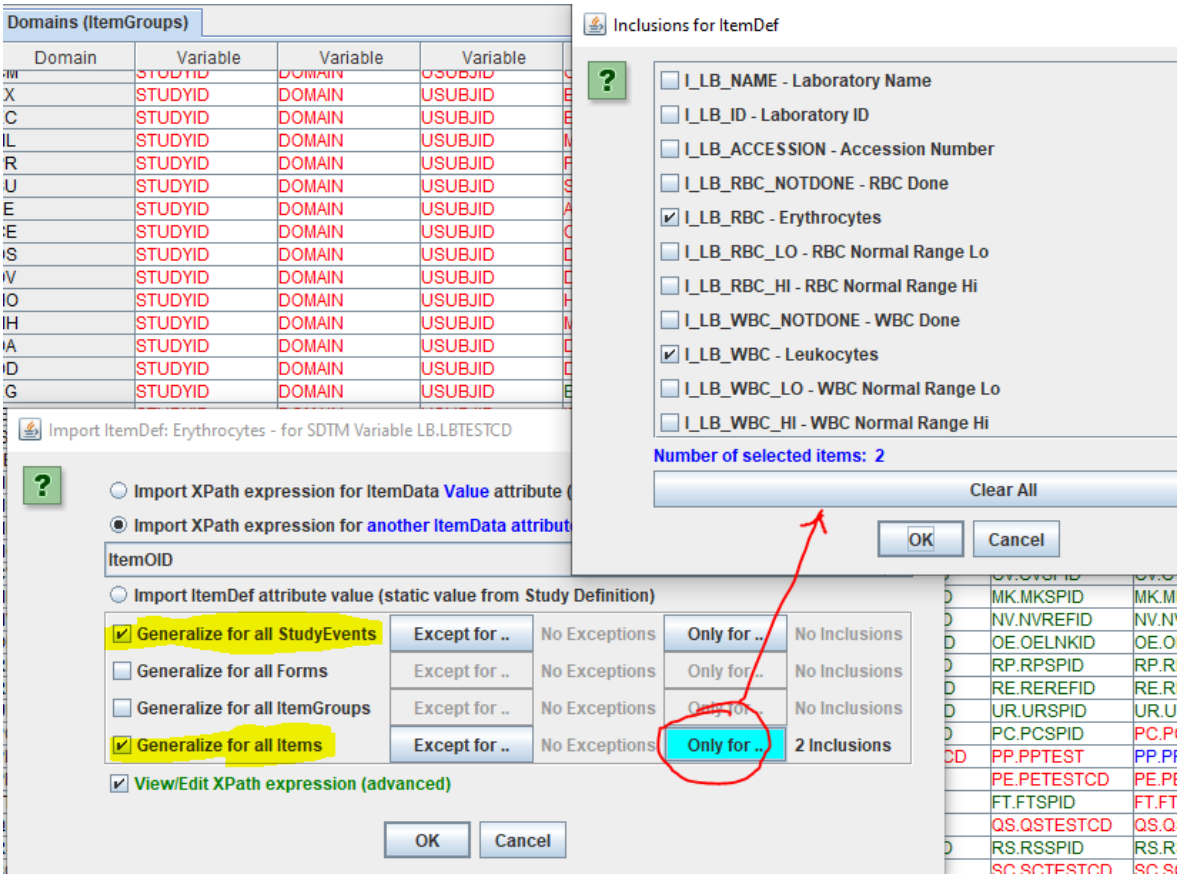
Now we want to filter out such data points. How do we do this?

Let's restart from the first step where we generate the mapping for LBTESTCD. Best is to delete all existing mappings by editing them by a double click on the cells that have a mapping and just delete the complete mapping script. We should then see our "original" state again in which the LBTESTCD, LBTEST and LBORRES SDTM cells are not "grayed out":

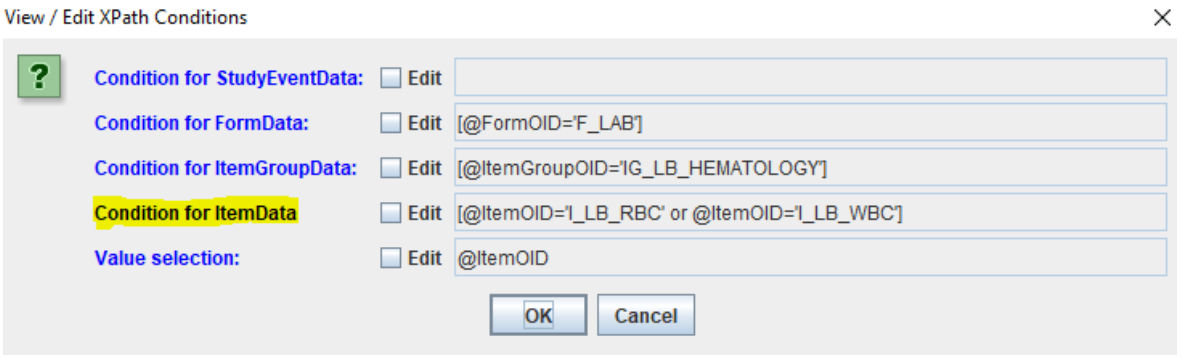
Alternative, we can right-click a cell in the "CES:LB" row, and then confirm that we want to remove the row:

and then drag-and-drop "LB" from the template again to the bottom, having the same effect.

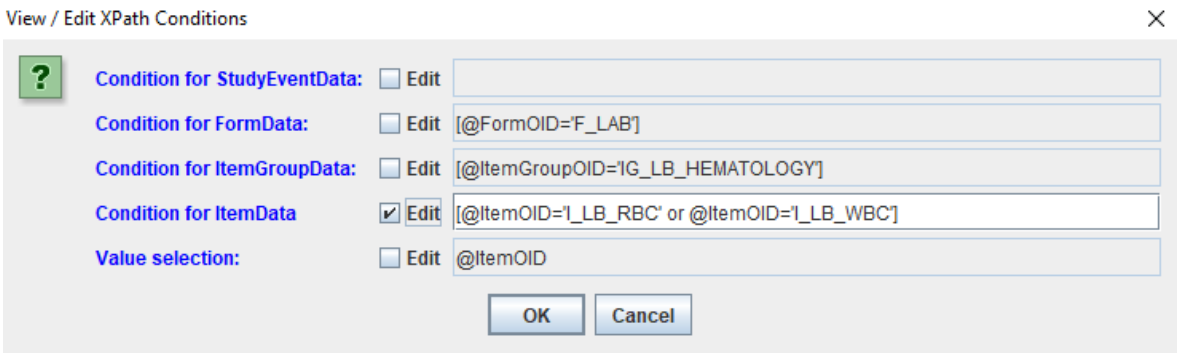
We then start the drag-and-drop from "Erythrocytes" to LBTESTCD again, do a "Generalize for all StudyEvents" again, with "Generalize for all Items" and "Only for ..." and selecting "Erythrocytes" and "Leukocytes":



taking care that also the checkbox "View/edit XPath expression …" is checked.
We then continue the process using the wizards as before, until we come to:



We now want to exclude data points for which the value is "UNKNOWN", so we check the checkbutton "Edit" for "Condition for ItemData", allowing us to edit the field on the right:



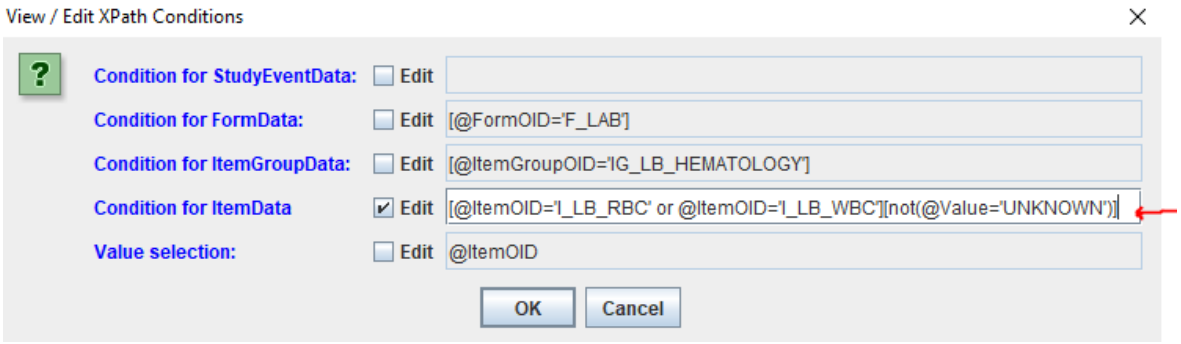
so that we can add an extra "conditions". This requires some basic knowledge of XPath which one can however quickly learn e.g. at the website "W3Schools".
Conditions ("predicates" in XPath language) go within square brackets, and as we want to exclude items for which the value is "UNKNOWN", we can e.g. write:

[not(@Value='UNKNOWN')]

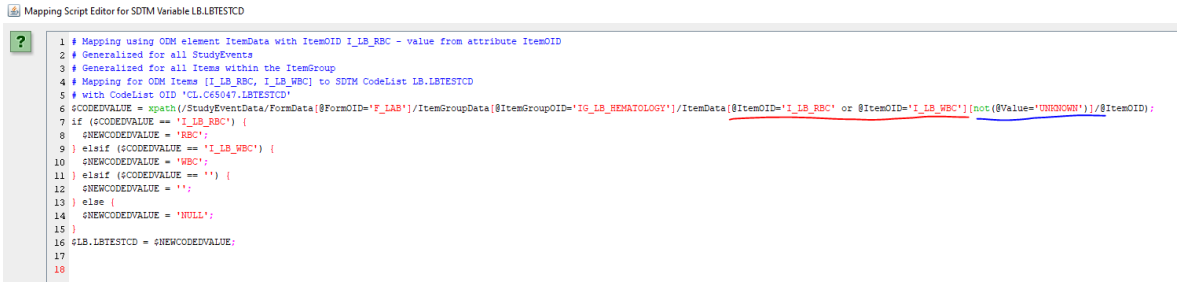
but we can also use:

[@Value != 'UNKNOWN']

if we use the former, we can just add it to the end of the XPath condition string for "Condition for ItemData":

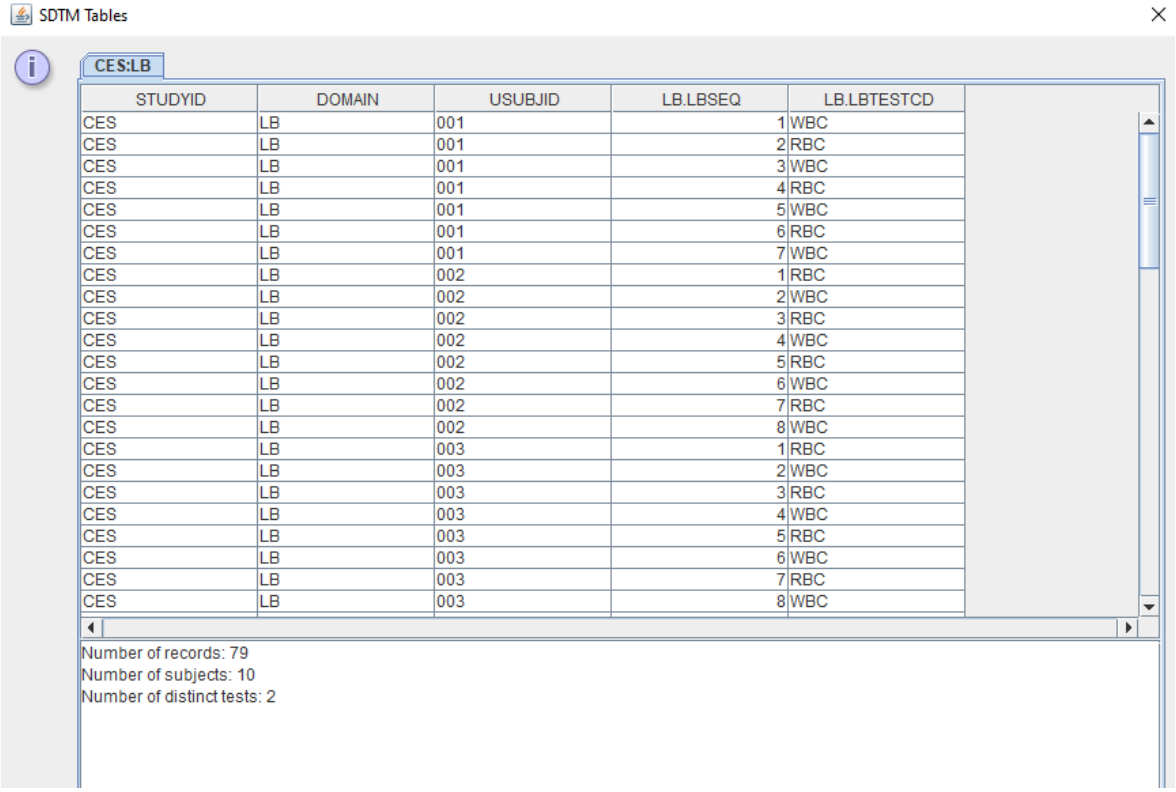


then clicking "OK" leads to the mapping script:

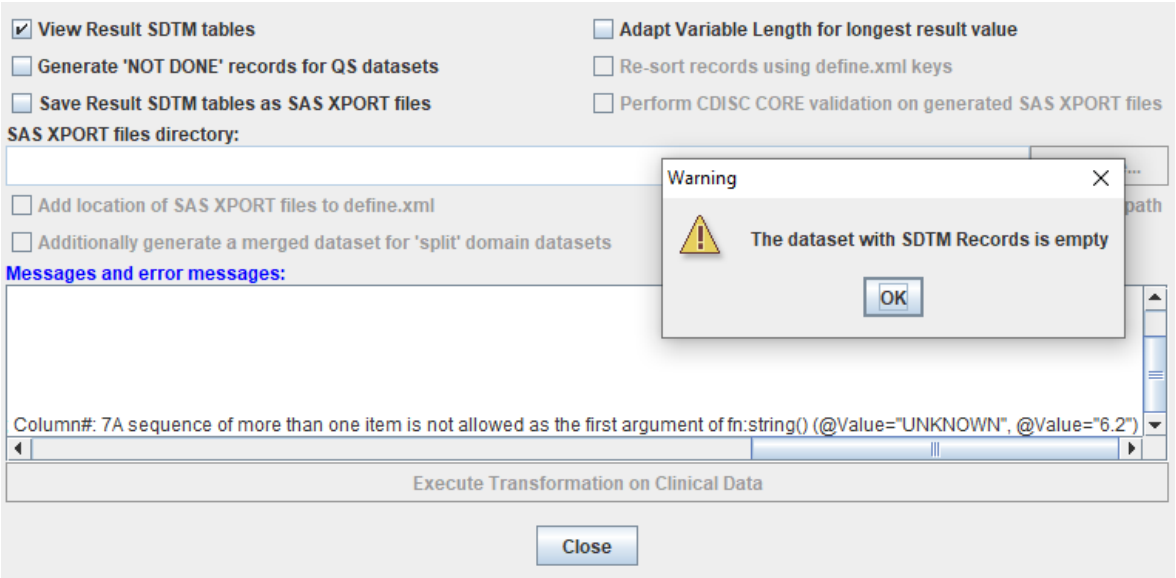


in which we see that we have 2 selection criteria for the item:
- the ItemOID (identifier) must either have the value "I_LB_RBC" or "I_LB_WBC"
- the value of the "Value" attribute may not be "UNKNOWN"

We can now already do a quick transformation run just to ensure that the XPath is a valid XPath (and nothing more), so after clicking "OK", we again use the "Transform" menu as before, ultimately leading to:

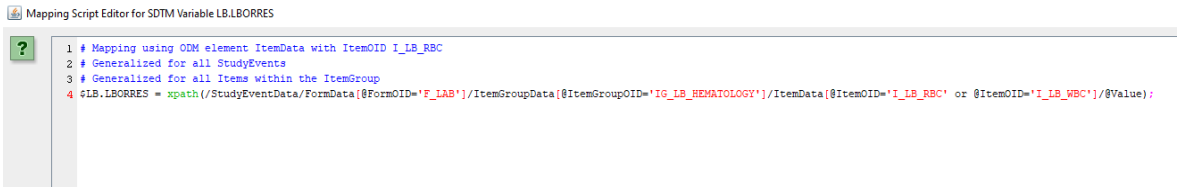


stating that this time, only 79 records have been generated (was 80 before …).
This however doesn't prove that our selection was right.
So we now do repeat generating the mappings for LBTEST and LBORRES just as before.
After having done so, executing the transformation again leads to an error:

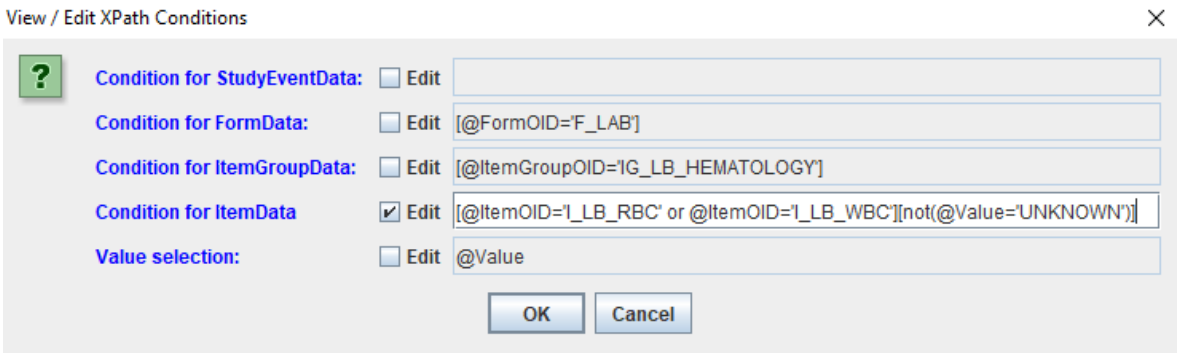


What went wrong?

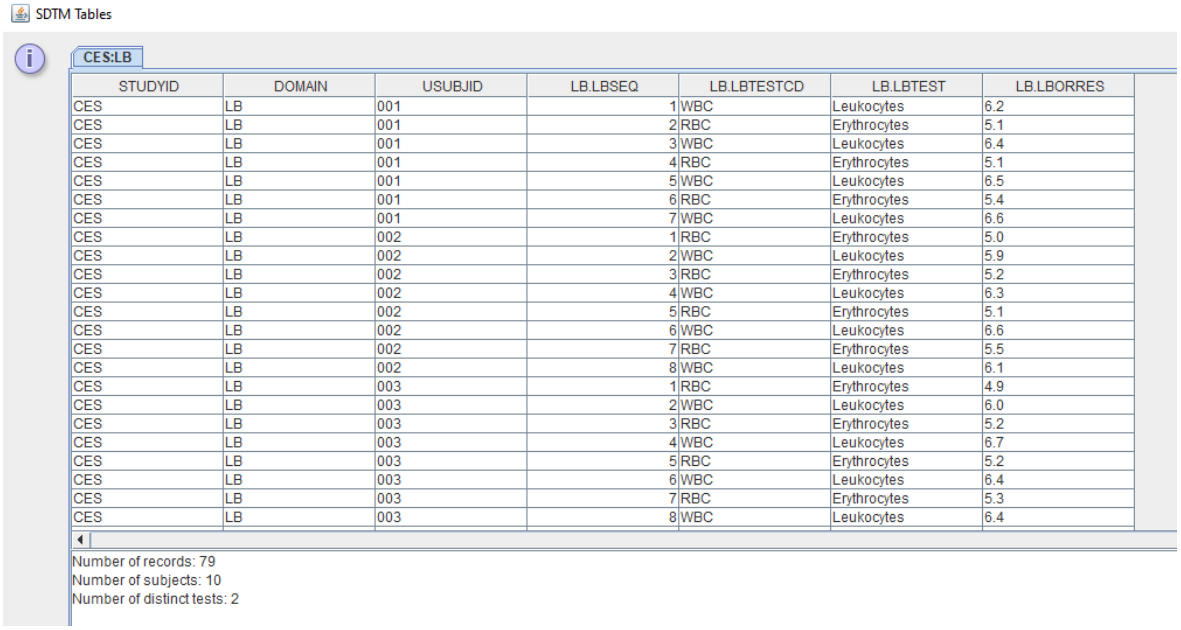
The reason is that for the mapping for LBORRES, we have:



where we do not have the additional condition for the value not being "UNKNOWN".
So if we do the drag-and-drop again to **LBORRES**, and allow to overwrite the existing one, we need to take care that the condition is also added there, i.e.:



and when then executing the mapping again, we get:



STUDYID	DOMAIN	USUBJID	LB.LBSEQ	LB.LBTESTCD	LB.LBTEST	LB.LBORRES
CES	LB	001	1	WBC	Leukocytes	6.2
CES	LB	001	2	RBC	Erythrocytes	5.1
CES	LB	001	3	WBC	Leukocytes	6.4
CES	LB	001	4	RBC	Erythrocytes	5.1
CES	LB	001	5	WBC	Leukocytes	6.5
CES	LB	001	6	RBC	Erythrocytes	5.4
CES	LB	001	7	WBC	Leukocytes	6.6
CES	LB	002	1	RBC	Erythrocytes	5.0
CES	LB	002	2	WBC	Leukocytes	5.9
CES	LB	002	3	RBC	Erythrocytes	5.2
CES	LB	002	4	WBC	Leukocytes	6.3
CES	LB	002	5	RBC	Erythrocytes	5.1
CES	LB	002	6	WBC	Leukocytes	6.6
CES	LB	002	7	RBC	Erythrocytes	5.5
CES	LB	002	8	WBC	Leukocytes	6.1
CES	LB	003	1	RBC	Erythrocytes	4.9
CES	LB	003	2	WBC	Leukocytes	6.0
CES	LB	003	3	RBC	Erythrocytes	5.2
CES	LB	003	4	WBC	Leukocytes	6.7
CES	LB	003	5	RBC	Erythrocytes	5.2
CES	LB	003	6	WBC	Leukocytes	6.4
CES	LB	003	7	RBC	Erythrocytes	5.3
CES	LB	003	8	WBC	Leukocytes	6.4

Number of records: 79
Number of subjects: 10
Number of distinct tests: 2

and we see that the data point with value "UNKNOWN" has now been excluded.

Excluding subjects

In a similar way, we can also exclude subjects. This however requires a bit of knowledge about the ODM structure.

In the ODM tree of the clinical data, the hierarchy is:

- ClinicalData
    - SubjectData
        - StudyEventData
            - FormData
                - ItemGroupData
                    - ItemData

Starting from "ClinicalData", each element has an attribute which has the same name, but with "Data" replaced by either "Key" or "OID" as the identifier
So we have "SubjectKey", "StudyEventOID", "FormOID", "ItemGroupOID", "ItemOID".

So, for an individual data point, the XPath expression will essentially be in the form:

/ClinicalData/SubjectData[@SubjectKey="..:"]/StudyEventData[@StudyEventOID="…"]/FormData[@FormOID="…"]/ItemGroupData[@ItemGroupOID="…"]/ItemData[@ItemOID="…"]/…

which we have probably already recognized in the xpath(.....) functions we have found in the mapping scripts.
As in SDTM-ETL, we always iterate over the subjects first, the content of the xpath(…) function starts from the "StudyEventData".
So, for example, in order to get the first visit ("StudyEvent" in ODM), the XPath expression line will be like:

$VISIT = xpath(/StudyEventData[@StudyEventOID="BASELINE"])

and if we want to state "all visits", we simply omit the "condition" (predicate):

$VISIT = xpath(/StudyEventData)

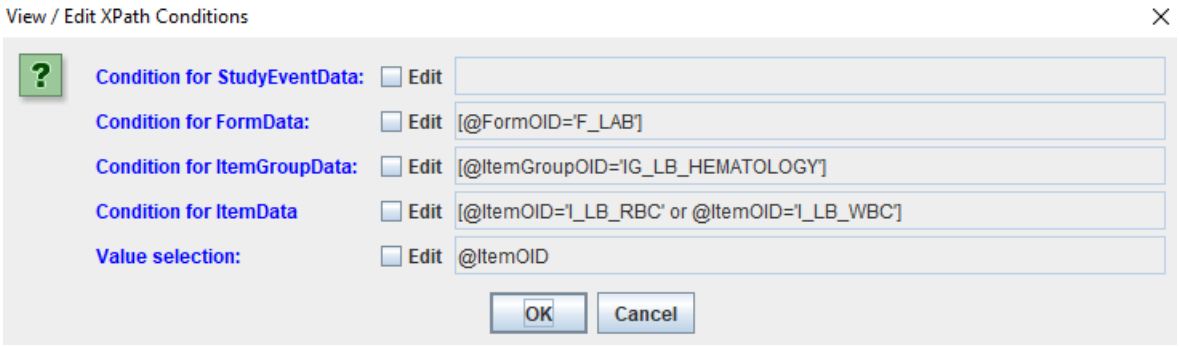
To navigate between the "tree nodes", i.e. between the different levels in the ODM tree, we use the following rules[4]:

..	go one level up
./	start from the current level

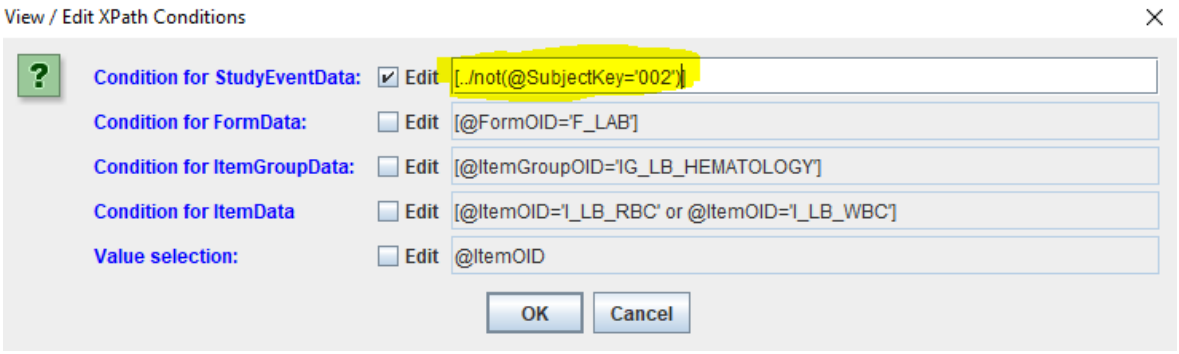
So, in order to select a subject, we need to add a condition that goes up from the "StudyEventData" level to the "SubjectData" level, and then take the value of the "SubjectKey" attribute. Lets have a try to exclude the subjects with the SubjectKey="002".

If we start from scratch again, instantiate a "study-specific domain", and drag-and-drop from Erythrocytes again, go through the wizards, check "View/edit XPath", we come to:

[4] These look very like what Linux/Unix users and Windows users that use the command line already know



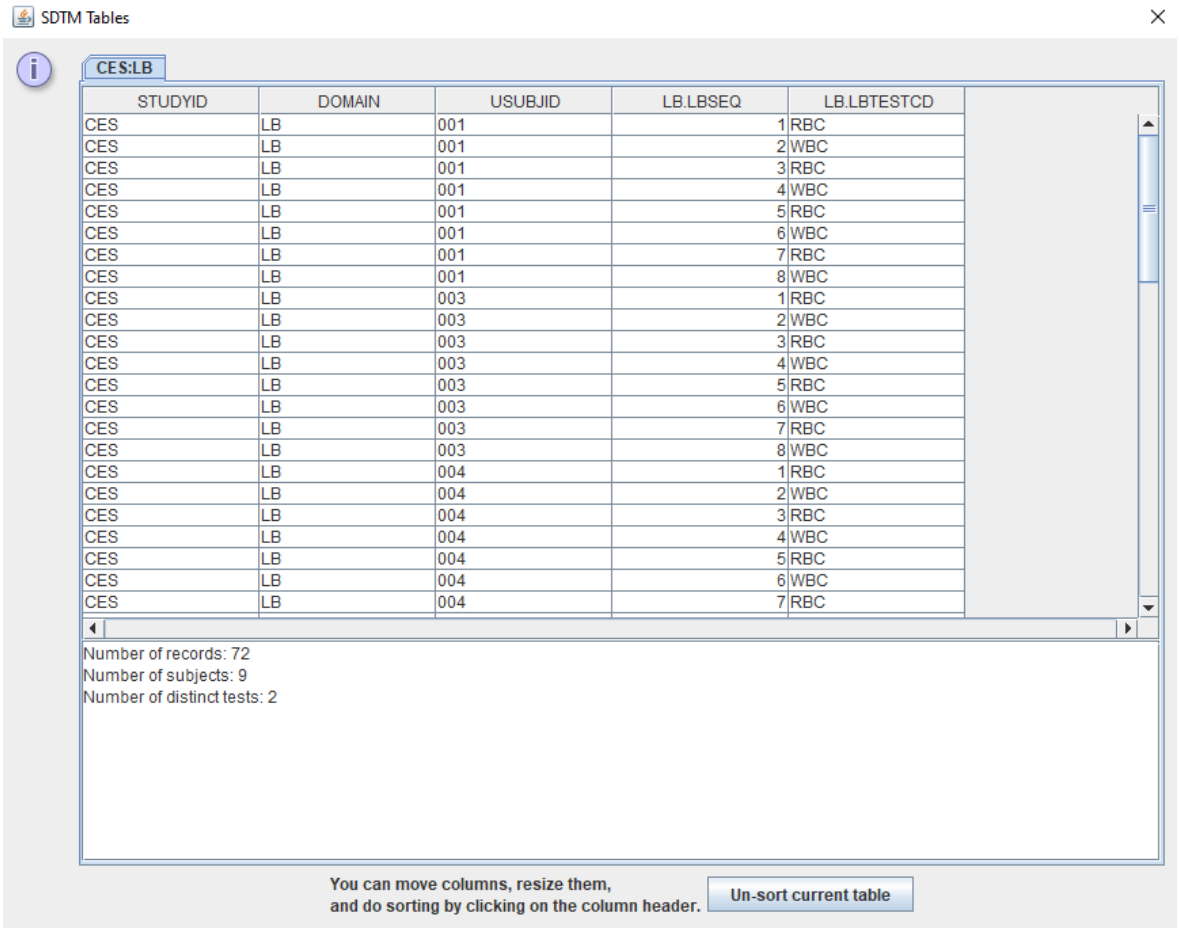
If we now want to exclude subject "002", we need to add a condition at the "StudyEventData" level, but need to go up one level up to "SubjectData", meaning we need to use "..".
So the condition for the exclusion becomes:



so that the left part of the mapping script for LBTESTCD becomes:

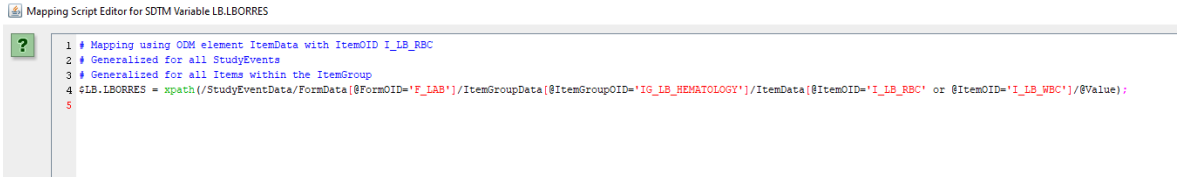
```
1  # Mapping using ODM element ItemData with ItemOID I_LB_RBC - value from attribut
2  # Generalized for all StudyEvents
3  # Generalized for all Items within the ItemGroup
4  # Mapping for ODM Items [I_LB_RBC, I_LB_WBC] to SDTM CodeList LB.LBTESTCD
5  # with CodeList OID 'CL.C65047.LBTESTCD'
6  $CODEDVALUE = xpath(/StudyEventData[../not(@SubjectKey='002')]/FormData[@FormOID
7  if ($CODEDVALUE == 'I_LB_RBC') {
8    $NEWCODEDVALUE = 'RBC';
9  } elsif ($CODEDVALUE == 'I_LB_WBC') {
10   $NEWCODEDVALUE = 'WBC';
11 } elsif ($CODEDVALUE == '') {
12   $NEWCODEDVALUE = '';
13 } else {
14   $NEWCODEDVALUE = 'NULL';
15 }
16 $LB.LBTESTCD = $NEWCODEDVALUE;
```

and when executing the mappings, leading to the result:

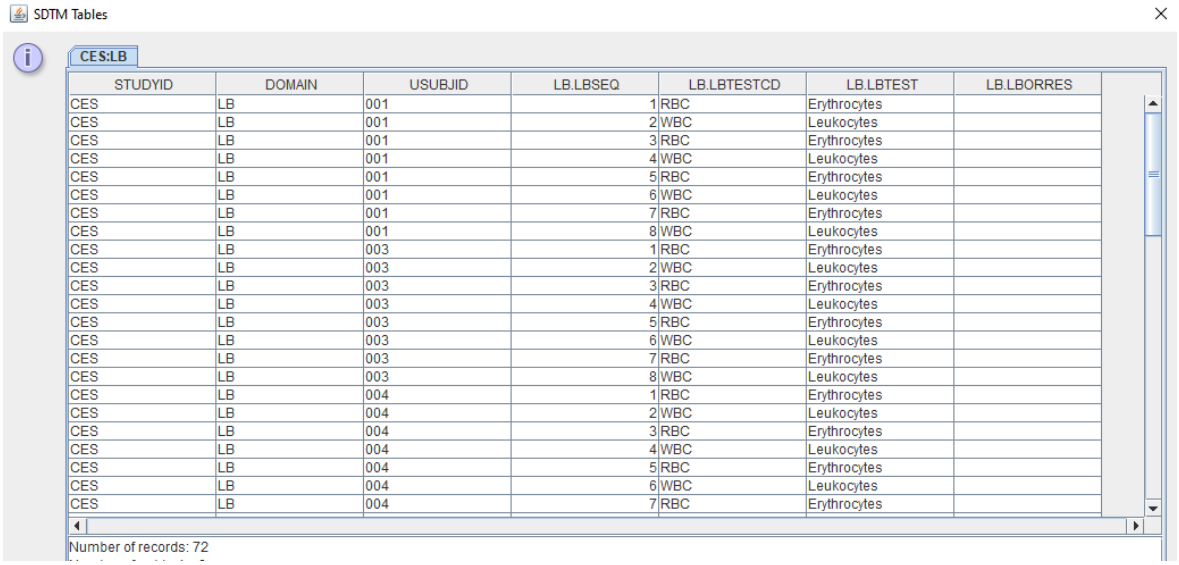


and we see that no data for subject "002" is present in the SDTM.

We can then also add the mappings for LBTEST and LBORRES by drag-and-drop, with the mapping script for e.g. LBORRES being:

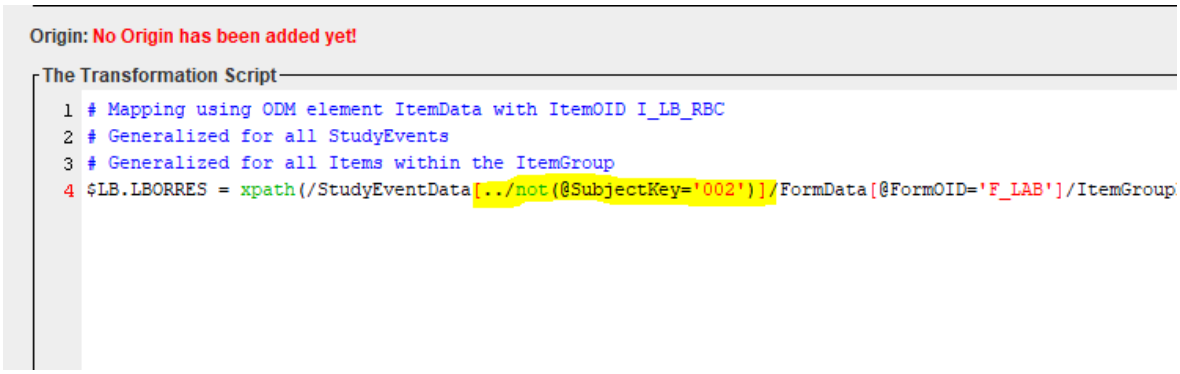


and when executing the mappings we get:

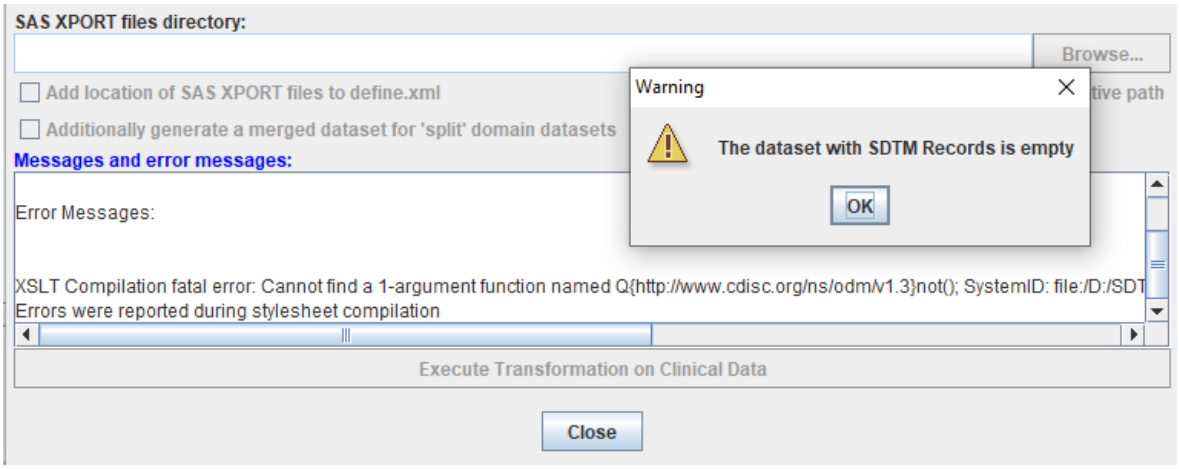


This time, we do not get an error, but there is no value for LBORRES either.

We can also add the condition to the mapping for LBORRES as:



but when then executing the mappings, we get an error:



The reason is that the system has a problem comparing the XPath expression for LBTESTCD (the "looping variable"), and the one for LBORRES, and transforming the latter to a relative path[5].

[5] This is currently being fixed in version 5.2

An introduction to relative XPath-s

We do not want to provide a full XPath tutorial here, but just some basics. For a more advanced tutorial on XPath we can recommend the W3C Schools tutorial. We will also limit ourselves to "relative" XPaths, as these are the only ones we will use.

In SDTM-ETL, in order to generate "one record per observation", determined as the "looping variable", which is usually defined as the "topic variable", i.e. --TERM, --TRT, or --TESTCD, (for SV this usually is VISITNUM - we we want to have "one record per subject per visit"), all other XPath expressions are recalculated by the software as relative against the XPath for the looping/topic variable. In some cases of a very complex XPath for the looping/topic variable, this may lead to problems. In such seldom cases, we may want to use a "relative XPath" for any of the other variables.
"Relative" XPath-s always start with either "./" or "../", or a combination of both.

Relative XPath starting with ...	Meaning
./	Stay on the current tree node, and do something on this node
./@xxx	Stay on the current tree node, and get the value of the xxx attribute For example: ./@Value means that we take the value of the attribute with the name "Value" (XPath is case-sensitive)
../	Go one level up in the ODM tree. For example: if we are on the ODM "ItemData" level, go one level up to the "ItemGroupData" level
../../	Go two levels up in the ODM tree. For example, if we are on the ODM "ItemData" level, go two levels up, up to the ODM "FormData" level
../../XXX	Go two levels up in the ODM tree and then go down again to the element with name "XXX". For example, for "../../ItemGroupData", if we are on the ODM "ItemData" level, go two levels up, up to the ODM "FormData" level, and then go down again to an "ItemGroupData" element. Usually, this will be followed by a "where-statement", which is called "predicate" in XPath language
[.....]	Everything between square brackets in XPath is a "predicate", so essentially a "where-statement" or "filter". Predicates are always used in combination with paths themselves. For example, if we are on the ODM "ItemData" level, and we have ../../ItemGroupData[@ItemGroupOID='XXX'], this means: go two levels up to the "FormData" level, then go down again to the "ItemGroupData" that has the value "XXX" for the ItemGroupOID attribute.

We will now explain some usual scenarios where we may want to use a relative XPath expression in case the one generated by the "drag-and-drop" gives unexpected results. This should however be very seldom. We will start with the case that the looping/topic variable

leads us to the "ItemData" level. We can see this by that that the XPath expression ends with something like:

ItemData[@ItemOID='xxx']/@Value or ItemData[@ItemOID='xxx']/@ItemOID

where "xxx" is the identifier of the ItemData data point in the ODM tree.

One typical example is for generating a value for "VISITNUM".
In this case, we want to start by retrieving from the "StudyEventOID" of "StudyEventData", as "StudyEvent" is the ODM designation for "visit"[6].
As the hierarchy in ODM is: ***SubjectData - StudyEventData - FormData - ItemGroupData - ItemData***, and we are at the "ItemData" level, this means that we need to go up 3 levels, i.e., our relative XPath expression will start with: ../../../
and we then want to take the value of the "StudyEventOID" attribute, the relative XPath expression becomes: ../../../@StudyEventOID.
It can also be that we need the value of the "repeat-key", then the relative XPath expression becomes: ../../../@StudyEventRepeatKey.

Another typical example is that we need to get the value of another "ItemData" within the same ItemGroupData group (usually this is called a "sibling"). For example, if, within the same group, the date of the data collection is captured in the ItemData with ItemOID="IT.LABDATE", the relative XPath expression becomes
../ItemData[@ItemOID='IT.LABDATE']/@Value

It also may happen that we need information from within another "group", but with the same form. This means that we need to go up two levels up the the "FormData" level, and then go down to the "other" ItemGroupData and then to the ItemData level. For example, if the collection date was captured in the "Common" group in the same form, which often is something like a "header" of the form. The relative XPath may then e.g. be like:
../../ItemGroupData[@ItemGroupOID='IG.COMMON']/ItemData[@ItemOID='IT.DATE]/@Value

It will be seldom that one needs to get something from within another form or another visit, but also for this seldom case, one can develop relative XPath expressions. One will then first want to go up three levels to the StudyEvent Data, and then to "another" FormData, like:
../../../FormData[@FormOID='FO.xxx']/ItemGroupData[@ItemGroupOID='IG.yyy']/ItemData[@ItemOID='IT.zzz']/@Value

And in very very seldom cases, we may e.g. want to exclude subjects for specific data points, which means we need to go up to the "SubjectData" level. For example, if we want to exclude data from the subject "002", the relative XPath expression, when starting from the ´"ItemData" level, will be:
../../../../[not(@SubjectKey='002')]. Remark that this surely is "bad practice" as, if we need to exclude some subjects, we surely want to this already in the XPath of the looping/topic variable. For example, for LBTESTCD being the "looping" variable, and we may want to exclude subjects "002" and "004", we may have
$VS.VSTESTCD = xpath(/StudyEventData[../@SubjectKey='002' **or** ../@SubjectKey='004']/FormData[@FormOID=FO.VITALS]/ItemGroupOID[@ItemGroupOID='IG.VITALS']/ItemData[@ItemOID='IT.VSVALUE']/@Value);

---

[6] The reason that ODM uses "StudyEvent" and not "Visit" is that it wanted to emphasize that this encompasses more that classic "visits", i.e. it can also be a telephone contact, an extract from an EHR, etc..

Remark the "or" in the predicate. One can also have "and" but this will not often occur.

For the SV (Subject Visits) domain/dataset, the looping/topic variable usually is VISITNUM, which is 99% of the cases is derived from the "StudyEventData" element.
So the mapping for VISITNUM in SV will then look like:

```
# we first make a temporary variable using the "StudyEventOID"
$VISITOID = xpath(/StudyEventData/@StudyEventOID);
# from which we derive a "number", e.g.
if($VISITOID='SE.BASE') {
   $SV.VISITNUM = 1;
} elsif($VISITOID='SE.FIRSTTREAT') {
   $SV.VISITNUM = 2;
…
```

If we then want to e.g. retrieve the visit date in SVDTC from a data point in the form "FO.VITALS" in the group "IG.COMMON" in the data point "IT.DATE", we may want to use:

```
$SV.SVDTC =
xpath(./FormData[@FormOID='FO.VITALS']/ItemGroupData[@ItemGroupOID='IG.COMMON']/ItemData[@ItemOID='IT.DATE']/@Value
```

with the initial "./" meaning: start at the level of the looping/topic variable, which is "StudyEventData", and then go down.
If we are at the StudyEvenData level, and we e.g. want to also retrieve the value of the "StudyEventRepeatKey", we can use the relative XPath

```
$VISITREPEATKEY = xpath(./@StudyEventRepeatKey);
```

Other types of "predicates" that can be of use are:

* [first()] : selects the first node, e.g /StudyEventData[first()] selects the first encounter "StudyEventData" node in the ODM tree. Remark that this will only select the earliest (in time) visit when the data in the ODM is in chronological order[7].
* [1] : this is equivalent to [first()]
* [last()] : selects the last node
* [last()-1] : selects the second last node

IMPORTANT REMARK: In most cases (>95% of the studies), you will <u>never</u> need to write relative paths yourself. Only if you encounter unexpected results (like VISITNUM being empty when your XPath generated by drag-and-drop is:
xpath(/StudyEventData/@StudyEventOID)).

Using relative XPath-s

We can however use a "relative path". As we are already at the "ItemData" level for the data points for the two lab tests, with the "looping variable" taking the ItemOID, all we need to state is that, instead of taking the ItemOID, we want to take the value of the "Value" attribute. So we use the relative XPath:

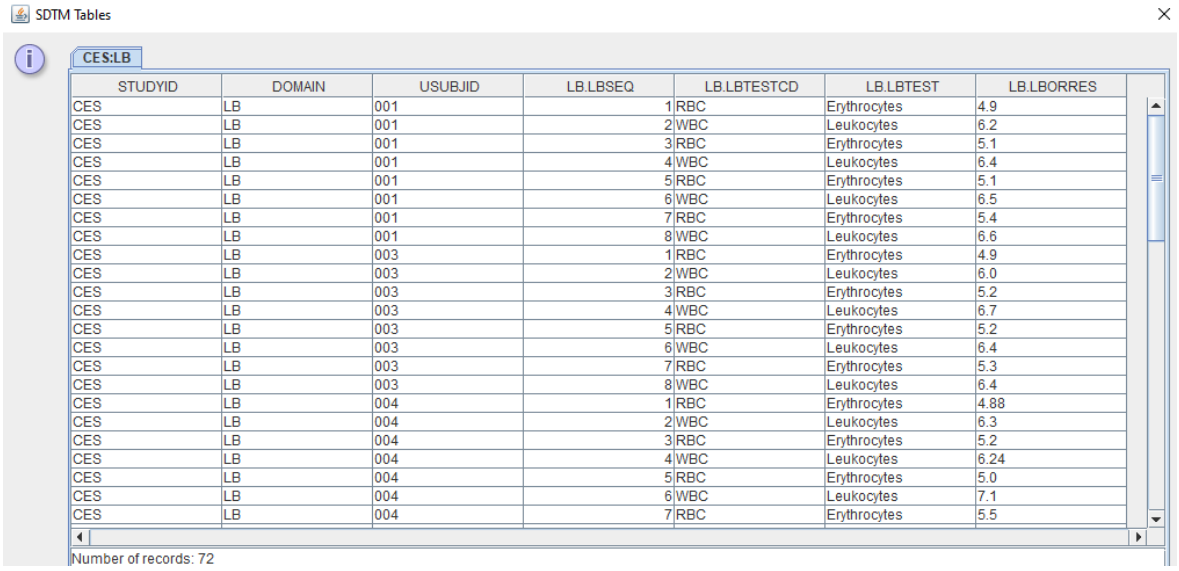
[7] This is essentially a requirement of ODM, but in practice ...

The Transformation Script

```
1 $LB.LBORRES = xpath(./@Value);
```

stating "at the current node, take the value of the 'Value' attribute".

The result then becomes:



STUDYID	DOMAIN	USUBJID	LB.LBSEQ	LB.LBTESTCD	LB.LBTEST	LB.LBORRES
CES	LB	001	1	RBC	Erythrocytes	4.9
CES	LB	001	2	WBC	Leukocytes	6.2
CES	LB	001	3	RBC	Erythrocytes	5.1
CES	LB	001	4	WBC	Leukocytes	6.4
CES	LB	001	5	RBC	Erythrocytes	5.1
CES	LB	001	6	WBC	Leukocytes	6.5
CES	LB	001	7	RBC	Erythrocytes	5.4
CES	LB	001	8	WBC	Leukocytes	6.6
CES	LB	003	1	RBC	Erythrocytes	4.9
CES	LB	003	2	WBC	Leukocytes	6.0
CES	LB	003	3	RBC	Erythrocytes	5.2
CES	LB	003	4	WBC	Leukocytes	6.7
CES	LB	003	5	RBC	Erythrocytes	5.2
CES	LB	003	6	WBC	Leukocytes	6.4
CES	LB	003	7	RBC	Erythrocytes	5.3
CES	LB	003	8	WBC	Leukocytes	6.4
CES	LB	004	1	RBC	Erythrocytes	4.88
CES	LB	004	2	WBC	Leukocytes	6.3
CES	LB	004	3	RBC	Erythrocytes	5.2
CES	LB	004	4	WBC	Leukocytes	6.24
CES	LB	004	5	RBC	Erythrocytes	5.0
CES	LB	004	6	WBC	Leukocytes	7.1
CES	LB	004	7	RBC	Erythrocytes	5.5

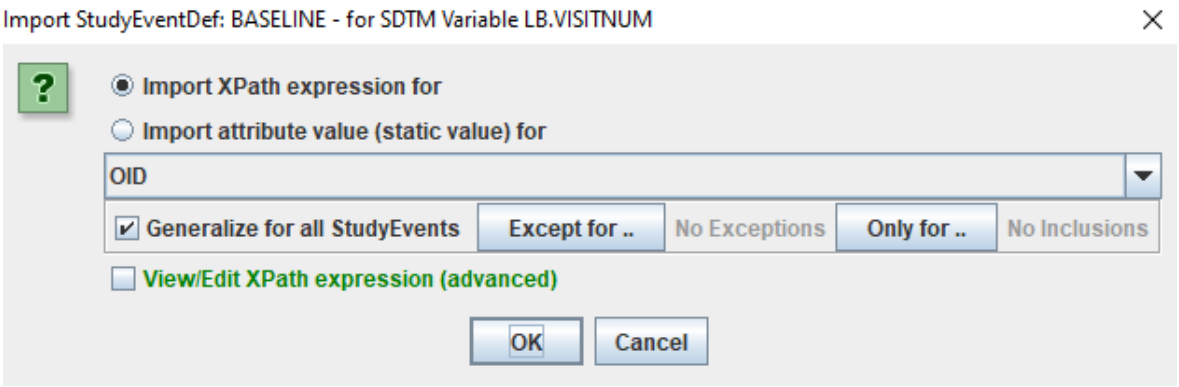
Number of records: 72

Again, subject 002 has been excluded.

REMARK: in version 4.6 there is a bug that causes an error when using this construct. This has been fixed in versions 5.0 and higher.

Let us now have a look at the VISITNUM and VISIT variables: we want to derive it from the "StudyEventOID" attribute of the "StudyEventData" element in the ODM tree.

If we just drag-and-drop from "StudyEvent: Baseline Visit" to the VISITNUM cell in the "CES:LB" row, the wizard for the first step is reduced to:

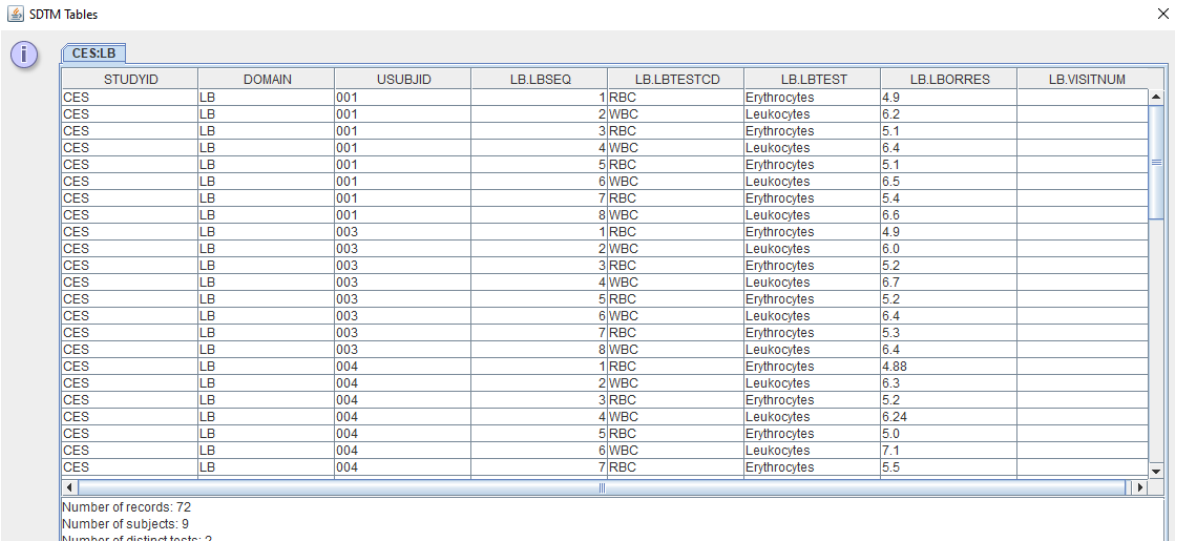


As we want to have it for all the visits in the study, we keep the checkbox "View/Edit XPath expression …" checked. Clicking "OK" then leads to the mapping script:

The Transformation Script

```
1 # Mapping using ODM element StudyEventData using value from attribute StudyEventOID
2 # Generalized for all StudyEvents
3 $LB.VISITNUM = xpath(/StudyEventData/@StudyEventOID/);
4
```

which we later still want to adapt.

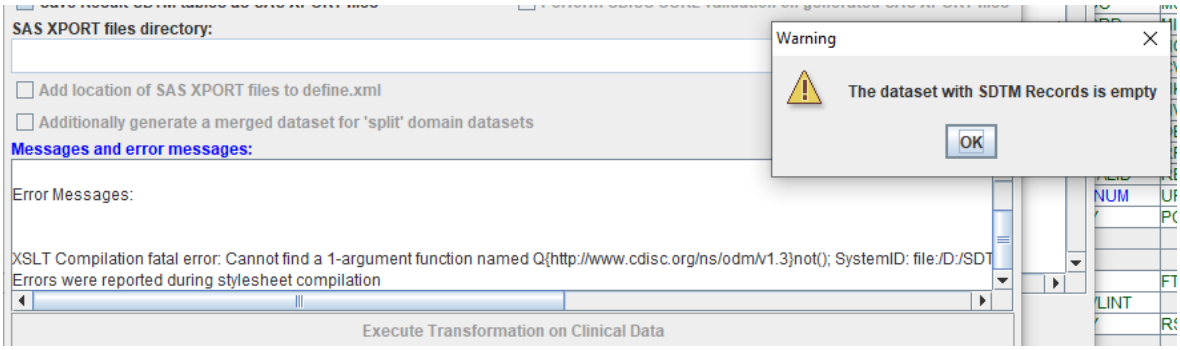


and we see the same incorrect behavior as we have seen for LBORRES before.
So we change the mapping script, either after drag-and-drop, or directly in the mapping script editor to:

The Transformation Script

```
1 # Mapping using ODM element StudyEventData using value from attribute StudyEventOID
2 # Generalized for all StudyEvents
3 $LB.VISITNUM = xpath(/StudyEventData[../not(@SubjectKey='002')]/@StudyEventOID/);
```

and when then executing the mappings:



we get the same error again.
IMPORTANT: Whether this error occurs for such cases depends on the version of the SDTM-ETL software. For example, in SDTM-ETL v.5.2, a better algorithm has been introduced for deriving the paths in the generated XSLT , together with some safeguards to

avoid such issues.

However, also here, we can use a "relative XPath". As the "looping variable" XPath goes down to the "Item" level, we need to go up 3 levels, and then take the value of the "StudyEventOID" element, i.e. we use:

The Transformation Script

```
# Mapping using ODM element StudyEventData using value from attribute StudyEventOID
# Generalized for all StudyEvents
$LB.VISITNUM = xpath(../../../@StudyEventOID);
```

and when then executing the mappings, the result is:

SDTM Tables

CES:LB

STUDYID	DOMAIN	USUBJID	LB.LBSEQ	LB.LBTESTCD	LB.LBTEST	LB.LBORRES	LB.VISITNUM
CES	LB	001	1	RBC	Erythrocytes	4.9	BASELINE
CES	LB	001	2	WBC	Leukocytes	6.2	BASELINE
CES	LB	001	3	RBC	Erythrocytes	5.1	WEEK_1
CES	LB	001	4	WBC	Leukocytes	6.4	WEEK_1
CES	LB	001	5	RBC	Erythrocytes	5.1	WEEK_1
CES	LB	001	6	WBC	Leukocytes	6.5	WEEK_1
CES	LB	001	7	RBC	Erythrocytes	5.4	WEEK_2
CES	LB	001	8	WBC	Leukocytes	6.6	WEEK_2
CES	LB	003	1	RBC	Erythrocytes	4.9	BASELINE
CES	LB	003	2	WBC	Leukocytes	6.0	BASELINE
CES	LB	003	3	RBC	Erythrocytes	5.2	WEEK_1
CES	LB	003	4	WBC	Leukocytes	6.7	WEEK_1
CES	LB	003	5	RBC	Erythrocytes	5.2	WEEK_1
CES	LB	003	6	WBC	Leukocytes	6.4	WEEK_1
CES	LB	003	7	RBC	Erythrocytes	5.3	WEEK_2
CES	LB	003	8	WBC	Leukocytes	6.4	WEEK_2
CES	LB	004	1	RBC	Erythrocytes	4.88	BASELINE
CES	LB	004	2	WBC	Leukocytes	6.3	BASELINE
CES	LB	004	3	RBC	Erythrocytes	5.2	WEEK_1
CES	LB	004	4	WBC	Leukocytes	6.24	WEEK_1
CES	LB	004	5	RBC	Erythrocytes	5.0	WEEK_1
CES	LB	004	6	WBC	Leukocytes	7.1	WEEK_1
CES	LB	004	7	RBC	Erythrocytes	5.5	WEEK_2

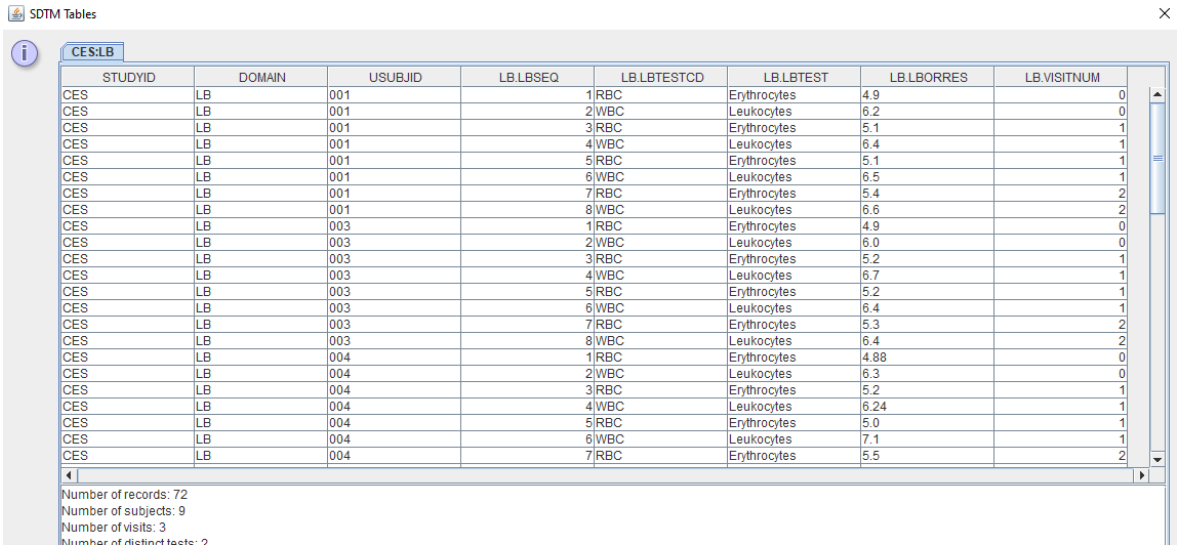
Number of records: 72

So here, it works better, and we also see that there is no data for subject 002, which is exactly what we want.
We can then further refine the script for "VISITNUM", e.g. as:

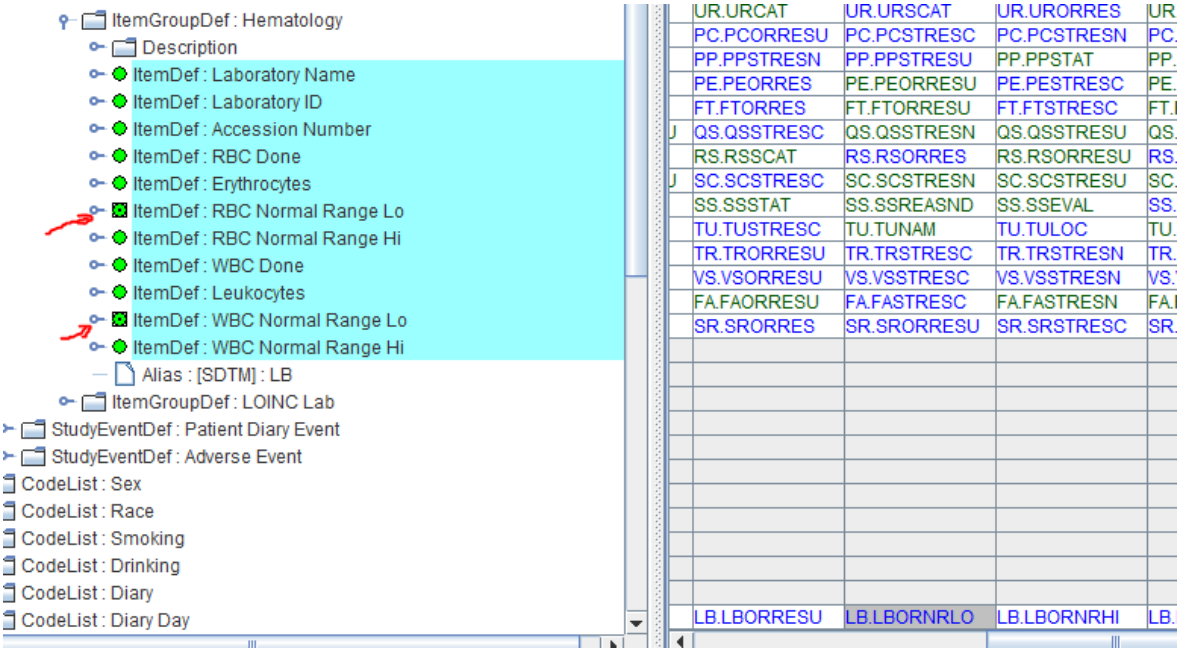
The Transformation Script

```
# Mapping using ODM element StudyEventData using value from attribute StudyEventOID
# Generalized for all StudyEvents
$TEMP = xpath(../../../@StudyEventOID);
if($TEMP = 'BASELINE') {
     $LB.VISITNUM = 0;
} elsif($TEMP = 'WEEK_1') {
     $LB.VISITNUM = 1;
} elsif($TEMP = 'WEEK_2') {
     $LB.VISITNUM = 2;
} else {
     $LB.VISITNUM = -999;
}
```

It is always a good idea to have an "else" item in such constructs to cover the unexpected case. For variables that are "numeric", one can e.g. use "-999" and for variables that are "character", one can e.g. use "TODO". If such unexpected case then occur, this will be visible in the result data.
After execution of the mappings, the result then is:



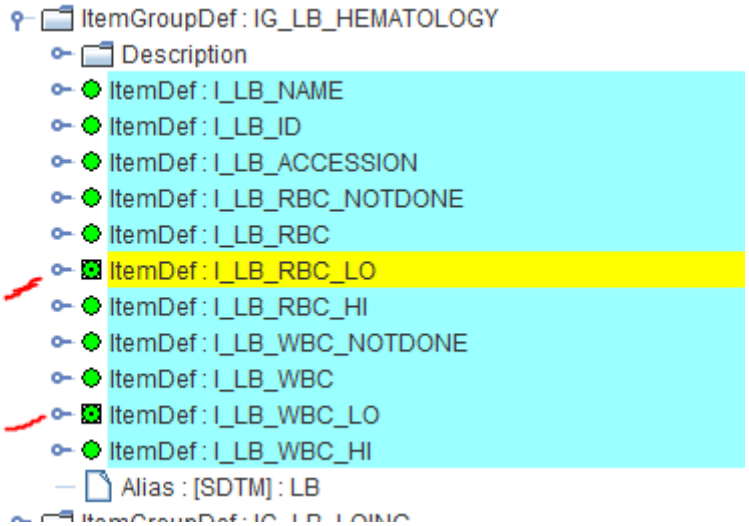
Let us now have a look at the SDTM variable LBORNRLO "Reference Range Lower Limit in Orig Unit"[8]. If the ODM, it maps to "RBC Normal Range Lo" and "WBC Normal Range Lo". As we have SDTM annotations in our ODM, when we click the cell "LBORNRLO", the ODM items for it are immediately marked as a "hot candidate":



We could now do drag-and-drop again, which we can already guess will lead to problems, but we can also use a "relative path" here. For that, we need to know the identifiers OIDs for "RBC Normal Range Lo" and "WBC Normal Range Lo". To find them, we can either hold the mouse over each of them, e.g.:

[8] The somewhat strange abbreviation for the "label" is due to the SDTM rule that "labels" may not be longer than 40 characters, which is a relict of SAS Transport 5.

or we can switch to the "OID" view using the menu "View - ODM tree with OIDs", leading to:



These ODM items are in the same group ("ItemGroup") as the items we are looping over, so all we need to do is to "go one level up" and then "go one level down" to either I_LB_RBC_LO or I_LB_WBC_LO.

A double-click on the cell LB.LBORNRLO allows us to start writing a mapping script that uses relative paths:

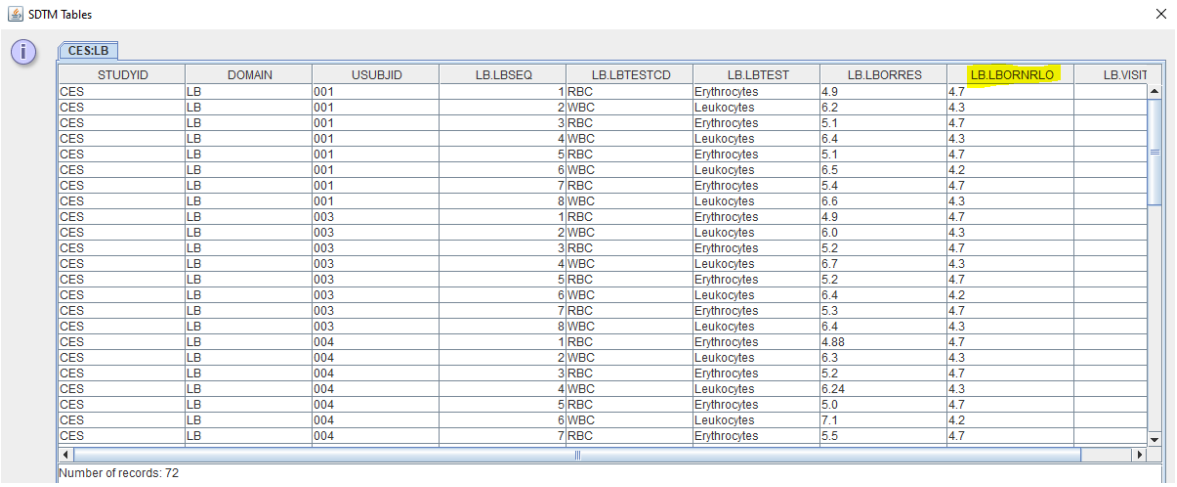
The Transformation Script

```
1  # Mapping script written manually
2  $RBC_LO = xpath(../ItemData[@ItemOID='I_LB_RBC_LO']/@Value);
3  $WBC_LO = xpath(../ItemData[@ItemOID='I_LB_WBC_LO']/@Value);
4  if($LB.LBTESTCD = 'RBC') {
5       $LB.LBORNRLO = $RBC_LO;
6  } elsif($LB.LBTESTCD = 'WBC') {
7       $LB.LBORNRLO = $WBC_LO;
8  } else {
9       $LB.LBORNRLO = 'TODO';
10 }
```

Notice that the XPath expressions need to come at the top, not within the if-elsif-else structure.
Some explanation:
- first we define relative paths for the **Value** of I_LB_RBC_LO (RBC Normal Range Lo) and for the **Value** of I_LB_WBC_LO (WBC Normal Range Lo) respectively

- we then set up an "if-elsif-else" structure
- stating that when the value of LBTESTCD is "RBC", we take the value from "RBC Normal Range Lo" and when the value of LBTESTCD is "WBC", we take the value from "WBC Normal Range Lo". This selection if of course important ...

The result then is:


SDTM Tables

CES:LB

STUDYID	DOMAIN	USUBJID	LB.LBSEQ	LB.LBTESTCD	LB.LBTEST	LB.LBORRES	LB.LBORNRLO	LB.VISIT
CES	LB	001	1	RBC	Erythrocytes	4.9	4.7	
CES	LB	001	2	WBC	Leukocytes	6.2	4.3	
CES	LB	001	3	RBC	Erythrocytes	5.1	4.7	
CES	LB	001	4	WBC	Leukocytes	6.4	4.3	
CES	LB	001	5	RBC	Erythrocytes	5.1	4.7	
CES	LB	001	6	WBC	Leukocytes	6.5	4.2	
CES	LB	001	7	RBC	Erythrocytes	5.4	4.7	
CES	LB	001	8	WBC	Leukocytes	6.6	4.3	
CES	LB	003	1	RBC	Erythrocytes	4.9	4.7	
CES	LB	003	2	WBC	Leukocytes	6.0	4.3	
CES	LB	003	3	RBC	Erythrocytes	5.2	4.7	
CES	LB	003	4	WBC	Leukocytes	6.7	4.3	
CES	LB	003	5	RBC	Erythrocytes	5.2	4.7	
CES	LB	003	6	WBC	Leukocytes	6.4	4.2	
CES	LB	003	7	RBC	Erythrocytes	5.3	4.7	
CES	LB	003	8	WBC	Leukocytes	6.4	4.3	
CES	LB	004	1	RBC	Erythrocytes	4.88	4.7	
CES	LB	004	2	WBC	Leukocytes	6.3	4.3	
CES	LB	004	3	RBC	Erythrocytes	5.2	4.7	
CES	LB	004	4	WBC	Leukocytes	6.24	4.3	
CES	LB	004	5	RBC	Erythrocytes	5.0	4.7	
CES	LB	004	6	WBC	Leukocytes	7.1	4.2	
CES	LB	004	7	RBC	Erythrocytes	5.5	4.7	

Number of records: 72

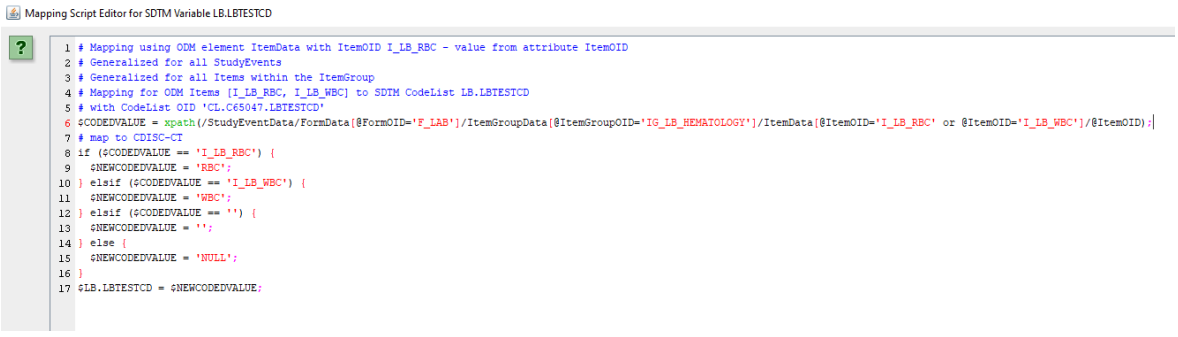
Using "relative paths" of course has the disadvantage that we need to write some mapping scripts themselves, and cannot always just rely on "drag-and-drop".
Therefore, there is also a more user-friendly mechanism: using "xpathfilter".

Using the "xpathfilter" utility (as of v.4.4)

A new feature introduced in SDTM-ETL v.4.6, and very well documented in the tutorial "Additional Filtering on Looping Variables" is the "xpathfilter" function.

It allows to do additional filtering after the having generated the XPath for the looping variable in the traditional way, mostly by simple drag-and-drop.

For our example to exclude data points having a value "UNKNOWN", we just drag-and-drop, follow the "wizards", and come to the following mapping script for the "looping variable" LBTESTCD:


Mapping Script Editor for SDTM Variable LB.LBTESTCD

```
1  # Mapping using ODM element ItemData with ItemOID I_LB_RBC - value from attribute ItemOID
2  # Generalized for all StudyEvents
3  # Generalized for all Items within the ItemGroup
4  # Mapping for ODM Items [I_LB_RBC, I_LB_WBC] to SDTM CodeList LB.LBTESTCD
5  # with CodeList OID 'CL.C65047.LBTESTCD'
6  $CODEDVALUE = xpath(/StudyEventData/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGroupOID='IG_LB_HEMATOLOGY']/ItemData[@ItemOID='I_LB_RBC' or @ItemOID='I_LB_WBC']/@ItemOID);
7  # map to CDISC-CT
8  if ($CODEDVALUE == 'I_LB_RBC') {
9    $NEWCODEDVALUE = 'RBC';
10 } elsif ($CODEDVALUE == 'I_LB_WBC') {
11   $NEWCODEDVALUE = 'WBC';
12 } elsif ($CODEDVALUE == '') {
13   $NEWCODEDVALUE = '';
14 } else {
15   $NEWCODEDVALUE = 'NULL';
16 }
17 $LB.LBTESTCD = $NEWCODEDVALUE;
```

So, nothing special, no adding additional "conditions" to the XPath expression either.

To do additional filtering, we can now use the "xpathfilter" function in a separate line (here line 7):

Mapping Script Editor for SDTM Variable LB.LBTESTCD

```
# Mapping using ODM element ItemData with ItemOID I_LB_RBC - value from attribute ItemOID
# Generalized for all StudyEvents
# Generalized for all Items within the ItemGroup
# Mapping for ODM Items [I_LB_RBC, I_LB_WBC] to SDTM CodeList LB.LBTESTCD
# with CodeList OID 'CL.C65047.LBTESTCD'
$CODEDVALUE = xpath(/StudyEventData/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGroupOID='IG_LB_HEMA
$CODEDVALUE = xpathfilter($CODEDVALUE,"not(@Value='UNKNOWN')");
# map to CDISC-CT
if ($CODEDVALUE == 'I_LB_RBC') {
  $NEWCODEDVALUE = 'RBC';
} elsif ($CODEDVALUE == 'I_LB_WBC') {
  $NEWCODEDVALUE = 'WBC';
} elsif ($CODEDVALUE == '') {
  $NEWCODEDVALUE = '';
} else {
  $NEWCODEDVALUE = 'NULL';
}
$LB.LBTESTCD = $NEWCODEDVALUE;
```

Notice that the result of "xpathfilter" must be stored in the same variable as where the xpath result is stored, here "$CODEDVALUE". Also please take care of correct use of double and single quotes, double quotes being used for stating that it is a string, and single quotes for the string condition in not(@Value='UNKNOWN');

The advantage is that one can then just mostly work with drag-and-drop, so that one does not need to repeat the exclusion condition to the script for each other variable.
So, for LBORRES, we can just drag-and-drop, accept the proposals made by the wizard, leading to:

Mapping Script Editor for SDTM Variable LB.LBORRES

```
# Mapping using ODM element ItemData with ItemOID I_LB_RBC
# Generalized for all StudyEvents
# Generalized for all Items within the ItemGroup
$LB.LBORRES = xpath(/StudyEventData/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGroupOID='IG_LB_HEMATOLOGY']/ItemData[@ItemOID='I_LB_RBC' or @ItemOID='I_LB_WBC']/@Value);
```

where we see that there is no "filtering" information is present, and also no manual relative XPath is needed.

For LBORNRLO (Reference Range Lower Limit-Std Units), we can also use drag-and-drop. but then need to do it twice, once for "Erythrocytes Normal Range Lo" and once for "Leukocytes Normal Range Lo". Please do not forget to uncheck "Generalize for all Items" in these cases. The partial generated script, with renaming local variables, then is like:

Mapping Script Editor for SDTM Variable LB.LBORNRLO

```
# Mapping using ODM element ItemData with ItemOID I_LB_RBC_LO
# Generalized for all StudyEvents
$RBC_LO = xpath(/StudyEventData/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGroupOID='IG_LB_HEMATOLOGY']/ItemData[@ItemOID='I_LB_RBC_LO']/@Value);
# Mapping using ODM element ItemData with ItemOID I_LB_WBC_LO
# Generalized for all StudyEvents
$WBC_LO = xpath(/StudyEventData/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGroupOID='IG_LB_HEMATOLOGY']/ItemData[@ItemOID='I_LB_WBC_LO']/@Value);

```

which does not need to contain any filtering information.

We can then continue differentiation between lower reference range for Erythrocytes and Leukocytes, e.g. like:

Mapping Script Editor for SDTM Variable LB.LBORNRLO

```
# Mapping using ODM element ItemData with ItemOID I_LB_RBC_LO
# Generalized for all StudyEvents
$RBC_LO = xpath(/StudyEventData/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGroupOID='IG_LB_HEMATOLOGY']/ItemData[@ItemOID='I_LB_RBC_LO']/@Value);
# Mapping using ODM element ItemData with ItemOID I_LB_WBC_LO
# Generalized for all StudyEvents
$WBC_LO = xpath(/StudyEventData/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGroupOID='IG_LB_HEMATOLOGY']/ItemData[@ItemOID='I_LB_WBC_LO']/@Value);
# differentiate between lower reference range for Erythrocytes and Leukocytes
if($LB.LBTESTCD = 'RBC') {
    $LB.LBORNRLO = $RBC_LO;
} elsif() {
    $LB.LBORNRLO = $WBC_LO;
} else {
    $LB.LBORNRLO = 'TODO';
}
```

when then executing the mapping scripts, the result is:

SDTM Tables

CES:LB

STUDYID	DOMAIN	USUBJID	LB.LBSEQ	LB.LBTESTCD	LB.LBORRES
CES	LB	001	1	WBC	6.2
CES	LB	001	2	RBC	5.1
CES	LB	001	3	WBC	6.4
CES	LB	001	4	RBC	5.1
CES	LB	001	5	WBC	6.5
CES	LB	001	6	RBC	5.4
CES	LB	001	7	WBC	6.6
CES	LB	002	1	RBC	5.0
CES	LB	002	2	WBC	5.9
CES	LB	002	3	RBC	5.2
CES	LB	002	4	WBC	6.3
CES	LB	002	5	RBC	5.1
CES	LB	002	6	WBC	6.6
CES	LB	002	7	RBC	5.5
CES	LB	002	8	WBC	6.1
CES	LB	003	1	RBC	4.9
CES	LB	003	2	WBC	6.0
CES	LB	003	3	RBC	5.2
CES	LB	003	4	WBC	6.7
CES	LB	003	5	RBC	5.2
CES	LB	003	6	WBC	6.4
CES	LB	003	7	RBC	5.3
CES	LB	003	8	WBC	6.4

Number of records: 79
Number of subjects: 10

so no need for adding XPath stuff for other variables than LBTESTCD.

We can then also simply drag-and-drop from "StudyEvent" in the ODM tree to "VISITNUM" in the SDTM table, checking "Generalize for all StudyEvents", leading to:

The Transformation Script

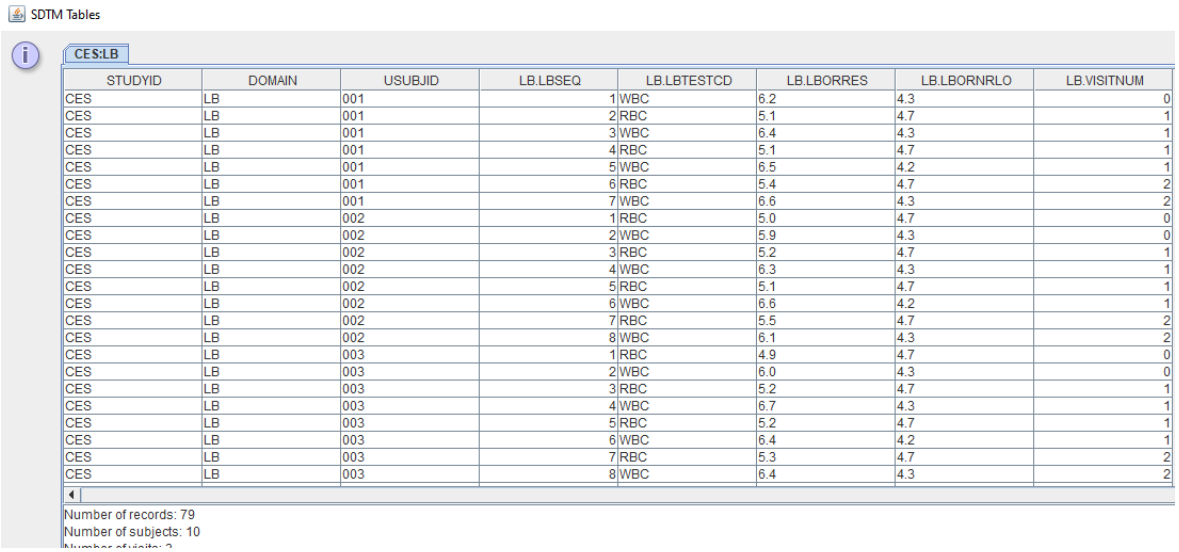
```
# Mapping using ODM element StudyEventData using value from attribute StudyEventOID
# Generalized for all StudyEvents
$LB.VISITNUM = xpath(/StudyEventData/@StudyEventOID/);
```

and then do the assignments, e.g.:

The Transformation Script

```
# Mapping using ODM element StudyEventData using value from attribute StudyEventOID
# Generalized for all StudyEvents
$STUDYEVENTOID = xpath(/StudyEventData/@StudyEventOID/);
if($STUDYEVENTOID = 'BASELINE') {
     $LB.VISITNUM = 0;
} elsif($STUDYEVENTOID = 'WEEK_1') {
     $LB.VISITNUM = 1;
} elsif($STUDYEVENTOID = 'WEEK_2') {
     $LB.VISITNUM = 2;
} else {
     $LB.VISITNUM = -999;
}
```

and after executing the mappings, the result is:



STUDYID	DOMAIN	USUBJID	LB.LBSEQ	LB.LBTESTCD	LB.LBORRES	LB.LBORNRLO	LB.VISITNUM
CES	LB	001	1	WBC	6.2	4.3	0
CES	LB	001	2	RBC	5.1	4.7	1
CES	LB	001	3	WBC	6.4	4.3	1
CES	LB	001	4	RBC	5.1	4.7	1
CES	LB	001	5	WBC	6.5	4.2	1
CES	LB	001	6	RBC	5.4	4.7	2
CES	LB	001	7	WBC	6.6	4.3	2
CES	LB	002	1	RBC	5.0	4.7	0
CES	LB	002	2	WBC	5.9	4.3	0
CES	LB	002	3	RBC	5.2	4.7	1
CES	LB	002	4	WBC	6.3	4.3	1
CES	LB	002	5	RBC	5.1	4.7	1
CES	LB	002	6	WBC	6.6	4.2	1
CES	LB	002	7	RBC	5.5	4.7	2
CES	LB	002	8	WBC	6.1	4.3	2
CES	LB	003	1	RBC	4.9	4.7	0
CES	LB	003	2	WBC	6.0	4.3	0
CES	LB	003	3	RBC	5.2	4.7	1
CES	LB	003	4	WBC	6.7	4.3	1
CES	LB	003	5	RBC	5.2	4.7	1
CES	LB	003	6	WBC	6.4	4.2	1
CES	LB	003	7	RBC	5.3	4.7	2
CES	LB	003	8	WBC	6.4	4.3	2

Number of records: 79
Number of subjects: 10

So one can say that using "xpathfilter" can considerably simplify generating the mappings without having to worry about additions to the XPath expressions in other variables than the looping variable, and no need for "relative XPath expressions".
One can however still use such "relative XPath expressions", e.g.:

The Transformation Script

```
# Mapping using ODM element StudyEventData using value from attribute StudyEventOID
# Generalized for all StudyEvents
# $STUDYEVENTOID = xpath(/StudyEventData/@StudyEventOID/);
$STUDYEVENTOID = xpath(../../../@StudyEventOID);
if($STUDYEVENTOID = 'BASELINE') {
     $LB.VISITNUM = 0;
} elsif($STUDYEVENTOID = 'WEEK_1') {
     $LB.VISITNUM = 1;
} elsif($STUDYEVENTOID = 'WEEK_2') {
     $LB.VISITNUM = 2;
} else {
     $LB.VISITNUM = -999;
}
```

leading to exactly the same result.

We can also give it a try for excluding a subject. The mapping script for the "looping variable" LBTESTCD then is:

The Transformation Script

```
# Mapping using ODM element ItemData with ItemOID I_LB_RBC - value from attribute ItemOID
# Generalized for all StudyEvents
# Generalized for all Items within the ItemGroup
# Mapping for ODM Items [I_LB_RBC, I_LB_WBC] to SDTM CodeList LB.LBTESTCD
# with CodeList OID 'CL.C65047.LBTESTCD'
$CODEDVALUE = xpath(/StudyEventData/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGroupOID='IG_LB_HEMATOLOGY']/
$CODEDVALUE = xpathfilter($CODEDVALUE,"not(../../../../@SubjectKey='002')");
# map to CDISC-CT
if ($CODEDVALUE == 'I_LB_RBC') {
  $NEWCODEDVALUE = 'RBC';
} elsif ($CODEDVALUE == 'I_LB_WBC') {
  $NEWCODEDVALUE = 'WBC';
} elsif ($CODEDVALUE == '') {
  $NEWCODEDVALUE = '';
} else {
  $NEWCODEDVALUE = 'NULL';
}
$LB.LBTESTCD = $NEWCODEDVALUE;
```

I.e. we go up 4 levels from the "ItemData" level up to the "SubjectData" level and then take the value of the "SubjectKey" attribute. The result is:

SDTM Tables

CES:LB

STUDYID	DOMAIN	USUBJID	LB.LBSEQ	LB.LBTESTCD	LB.LBORRES	LB.LBORNRLO
CES	LB	001	1	RBC	UNKNOWN	4.7
CES	LB	001	2	WBC	6.2	4.3
CES	LB	001	3	RBC	5.1	4.7
CES	LB	001	4	WBC	6.4	4.3
CES	LB	001	5	RBC	5.1	4.7
CES	LB	001	6	WBC	6.5	4.2
CES	LB	001	7	RBC	5.4	4.7
CES	LB	001	8	WBC	6.6	4.3
CES	LB	003	1	RBC	4.9	4.7
CES	LB	003	2	WBC	6.0	4.3
CES	LB	003	3	RBC	5.2	4.7
CES	LB	003	4	WBC	6.7	4.3
CES	LB	003	5	RBC	5.2	4.7
CES	LB	003	6	WBC	6.4	4.2
CES	LB	003	7	RBC	5.3	4.7
CES	LB	003	8	WBC	6.4	4.3
CES	LB	004	1	RBC	4.88	4.7
CES	LB	004	2	WBC	6.3	4.3
CES	LB	004	3	RBC	5.2	4.7
CES	LB	004	4	WBC	6.24	4.3
CES	LB	004	5	RBC	5.0	4.7
CES	LB	004	6	WBC	7.1	4.2
CES	LB	004	7	RBC	5.5	4.7

Number of records: 72

where we see that there are no records anymore for subject "002", but of course the value "UNKNOWN" for subject "001" appears again.

We can even accumulate such filters (use with care though), here to exclude as well subject "002" as values "UNKNOWN" for LBORRES. The script for LBTESTCD is:

The Transformation Script

```
# Mapping using ODM element ItemData with ItemOID I_LB_RBC - value from attribute ItemOID
# Generalized for all StudyEvents
# Generalized for all Items within the ItemGroup
# Mapping for ODM Items [I_LB_RBC, I_LB_WBC] to SDTM CodeList LB.LBTESTCD
# with CodeList OID 'CL.C65047.LBTESTCD'
$CODEDVALUE = xpath(/StudyEventData/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGroupOID='IG_LB_H
$CODEDVALUE = xpathfilter($CODEDVALUE,"not(../../../../@SubjectKey='002')");
$CODEDVALUE = xpathfilter($CODEDVALUE,"not(@Value='UNKNOWN')");
# map to CDISC-CT
if ($CODEDVALUE == 'I_LB_RBC') {
  $NEWCODEDVALUE = 'RBC';
} elsif ($CODEDVALUE == 'I_LB_WBC') {
  $NEWCODEDVALUE = 'WBC';
} elsif ($CODEDVALUE == '') {
  $NEWCODEDVALUE = '';
} else {
  $NEWCODEDVALUE = 'NULL';
}
$LB.LBTESTCD = $NEWCODEDVALUE;
```

with the result:

SDTM Tables

CES:LB

STUDYID	DOMAIN	USUBJID	LB.LBSEQ	LB.LBTESTCD	LB.LBORRES	LB.LBORNRLO
CES	LB	001	1	WBC	6.2	4.3
CES	LB	001	2	RBC	5.1	4.7
CES	LB	001	3	WBC	6.4	4.3
CES	LB	001	4	RBC	5.1	4.7
CES	LB	001	5	WBC	6.5	4.2
CES	LB	001	6	RBC	5.4	4.7
CES	LB	001	7	WBC	6.6	4.3
CES	LB	003	1	RBC	4.9	4.7
CES	LB	003	2	WBC	6.0	4.3
CES	LB	003	3	RBC	5.2	4.7
CES	LB	003	4	WBC	6.7	4.3
CES	LB	003	5	RBC	5.2	4.7
CES	LB	003	6	WBC	6.4	4.2
CES	LB	003	7	RBC	5.3	4.7
CES	LB	003	8	WBC	6.4	4.3
CES	LB	004	1	RBC	4.88	4.7
CES	LB	004	2	WBC	6.3	4.3
CES	LB	004	3	RBC	5.2	4.7
CES	LB	004	4	WBC	6.24	4.3
CES	LB	004	5	RBC	5.0	4.7
CES	LB	004	6	WBC	7.1	4.2
CES	LB	004	7	RBC	5.5	4.7
CES	LB	004	8	WBC	6.7	4.3

Number of records: 71
Number of subjects: 9
Number of visits: 3
Number of distinct tests: 2

Please note again that "xpathfilter" can currently only be used for "looping variables".

Conclusions

Filtering in SDTM-ETL on items (visits, forms, subforms, data points) from the ODM tree is done using XPath expressions that are usually generated automatically from the "drag-and-drop" and the following "wizards", especially using the "Generalize for" with "Except for …" and "Only for …" checkboxes and buttons.
Additional filtering can be achieved by adding predicates ("where-statements") using the GUI by selecting the "View/Edit XPath expression" checkbox.

In the first step of the mapping execution, all the XPath expressions are converted ("under the hood") to "relative" expressions relative to the "looping variable" which usually is the SDTM

"topic variable". In case the user has very special XPath expressions, this conversion ("under the hood") may, in seldom cases, lead to unexpected results. In such seldom cases, one may revert to adding such relative XPath-s itself. This tutorial explains how this can be done, and demonstrates some usual scenarios.
For the looping/topic variable, one may also add additional filtering using the "xpathfilter()" function, which is also explained in this tutorial.