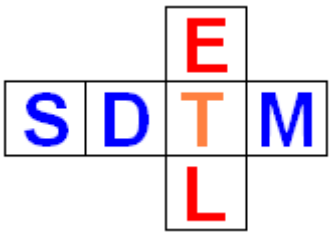
SDTM-ETL 5.x: Non-Standard Variables and Supplemental Qualifiers

Author: Jozef Aerts, XML4Pharma

Last update: **2026-08-03**



Introduction

Non-Standard Variables (NSVs) and Supplemental Qualifiers (SUPPQUALs) are all about data points that do not fit the traditional SDTM (or SEND) data structures. Both names are used almost as synonyms, the name "SUPPQUAL" coming from the data domain-like data structure of which the name starts with "SUPP". So, if we e.g. have a variable that is about demographics, but there is no SDTM-DM variable for it, then we will set up a "SUPPDM" dataset to keep the values for this NSV (and possibly for other NSVs for DM) in this SUPPDM dataset. Such SUPPxx datasets have a vertical structure, following the EAV (Entity-Attribute-Value) paradigm. See the SDTM-IGs for more details.

In SDTM-ETL, NSVs are treated just like any other "standard" variables. It is just at the very end of the process, when the mappings are executed, that the data for them is "split off" into a corresponding SUPPxx dataset.

In the last 10 years, the number of NSVs in submissions has considerably decreased, as CDISC has added "often used NSV" (one can say "most popular" NSVs to the standard domains as "standard variables".

For example, before SDTMIG-3.4, "--CLSIG" (Clinical Significance) was a very popular NSV, but became a standard variable in SDTMIG-3.4 which can be used in any "Findings" domain. Other typical NSVs are e.g. for storing the ATC code in "Prior and Concomitant Medications" and AESOSP (Other Medically Important SAE) and AETRTEM (AE Treatment Emergent).
Long time ago, it was even typical to store variable information that is nowadays stored in SC (Subject Characteristics) as NSVs in DM.

Essentially, NSV data should not go into SUPPxx datasets. It looks as that the reason that the

SDTMIGs state they MUST be split off into SUPPxx datasets is that the reviewers do not know and understand SDTM sufficiently to understand the difference between a "standard" and "non-standard" variable, this although since Define-XML 2.1, there is an attribute "IsNonStandard" for marking such variables. The latter would allow to e.g. have viewers with the NSVs remaining in the domain/dataset where they belong, but colored differently. The reviewers however do not seem to have access to such "smart viewers", although some of these are free and even "open source". It looks as reviewers can only use the "Universal SAS Viewer" or the old "SASViewer", which do not have such "smart features".

A simple example: multiple race

In this tutorial, we will demonstrate the use of NSVs and how they are typically and automatically "split off" in SUPPxx datasets for the case of "multiple race" in DM. For this, the SDTMIG-3.4 states:

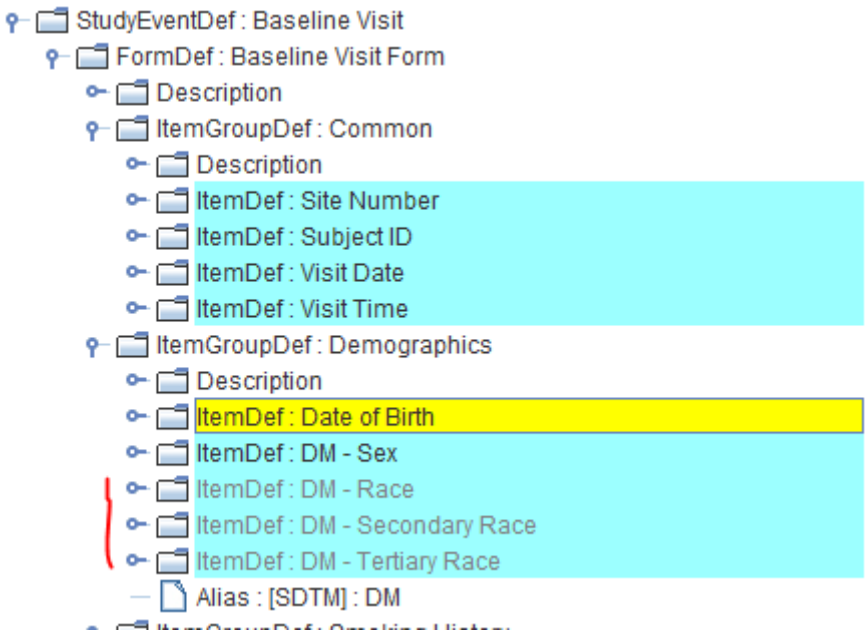
6. Submission of multiple race responses should be represented in the Demographics (DM) domain and Supplemental Qualifiers (SUPPDM) dataset as described in Section 4.2.8.3, Multiple Values for a Non-result Qualifier Variable. If multiple races are collected, then the value of RACE should be "MULTIPLE" and the additional information will be included in the Supplemental Qualifiers dataset. Controlled terminology for RACE should be used in both DM and SUPPDM so that consistent values are available for summaries regardless of whether the data are found in a column or row. If multiple races were collected and 1 was designated as primary, RACE in DM should be the primary race and additional races should be reported in SUPPDM. When additional free-text information is reported about subject's race using "Other, Specify", sponsors should refer to Section 4.2.7.1, "Specify" Values for Non-Result Qualifier Variables. If race was collected via an "Other, Specify" field and the sponsor chooses not to map the value as described in the current FDA guidance (see CDISC Notes for RACE in the domain specification), then the value of RACE should be "OTHER". For subjects who refuse to provide or do not know their race information, the value of RACE could be "UNKNOWN". See DM Example 4, DM Example 5, DM Example 6, and DM Example 7.

So essentially, there are 2 use cases: one where of the races has been specified as "primary" race, and one where one the provided races are equivalent. In the latter case, the value going into DM.RACE is "MULTIPLE" and the "equivalent" collected races go into RACE1, RACE2, RACE3, etc..
In the former case, where one of the races is marked as "primary", the value goes into the standard variable RACE, and the others (secondary, tertiary, …) go into the NSVs RACE1, RACE2, etc.

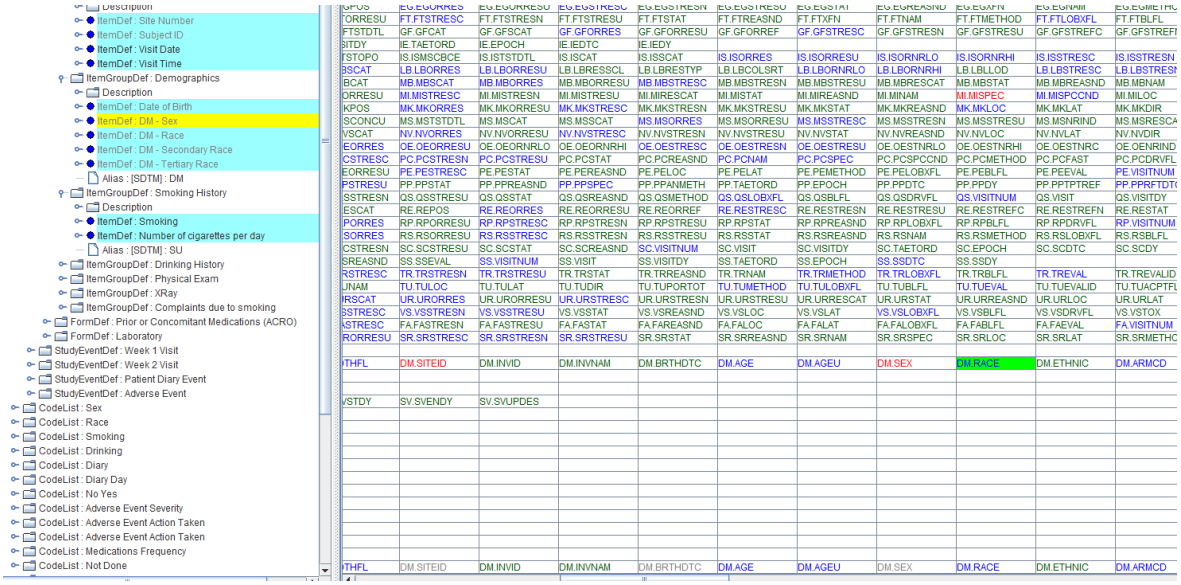
We will treat both cases separately, but first we will learn how to set up NSVs anyway.

Setting up the Non-Standard-Variables

First, let us have a look at our ODM structure:

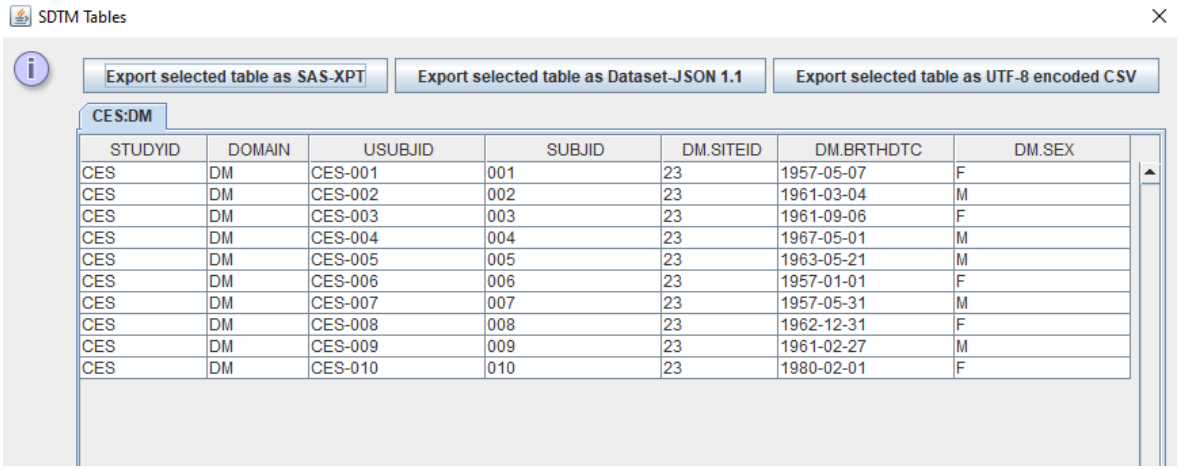


where we see that we have "Race", "Secondary Race" and "Tertiary Race". This means that we may need 2-3 NSVs (RACE1, RACE2, RACE3) in our DM depending on whether we have the first case (one race is the primary) and the second case (all races are equivalent). Suppose that we already started working on DM, and have already developed the mappings for SITE, SEX, BRTHDTC (date of birth) and DMDTC (Date/time of Collection), so our SDTM DM row shows up as:



On the left side (ODM side) we see that a number of items have been "grayed out", meaning that they have been used in at least one mapping.

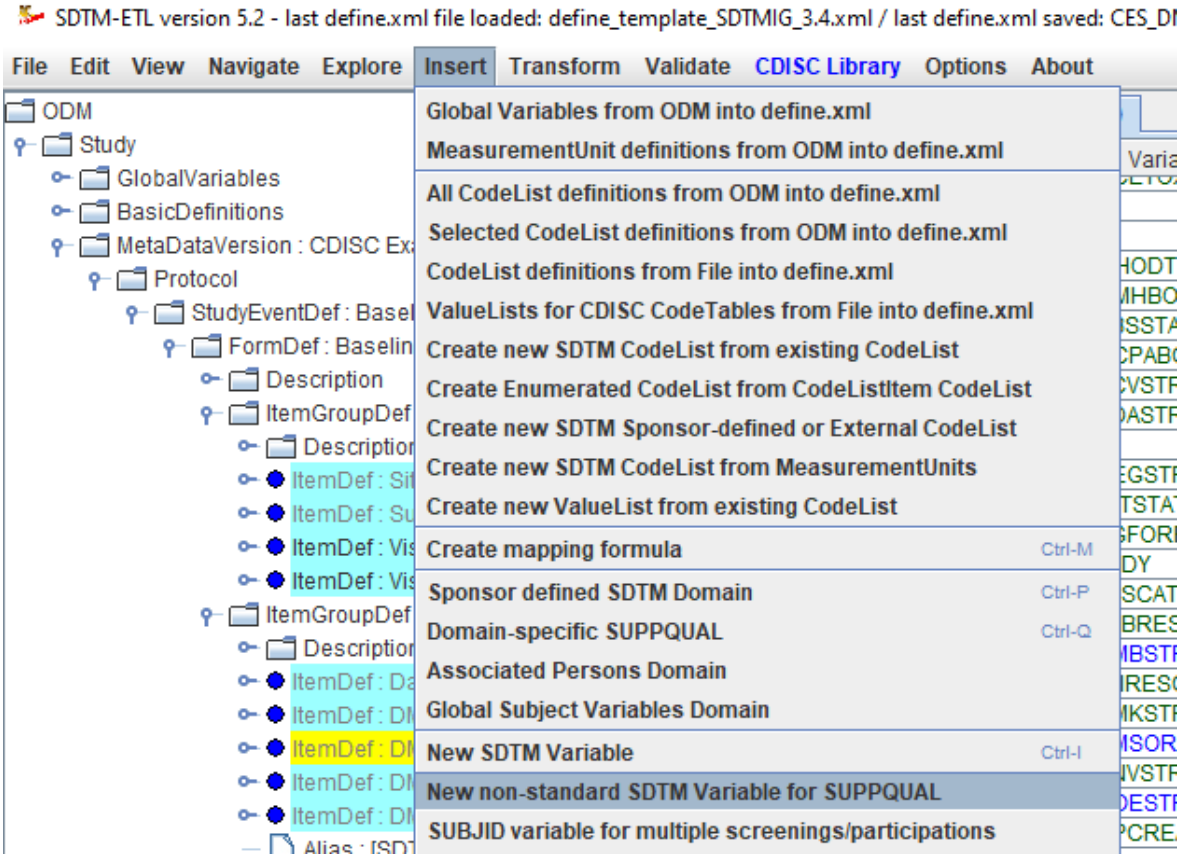
When we execute the mappings using the menu "Transform - Generate Transformation (XSLT) Code, followed by "Execute Transformation Code", provide the ODM file with the clinical data, this then leads to:



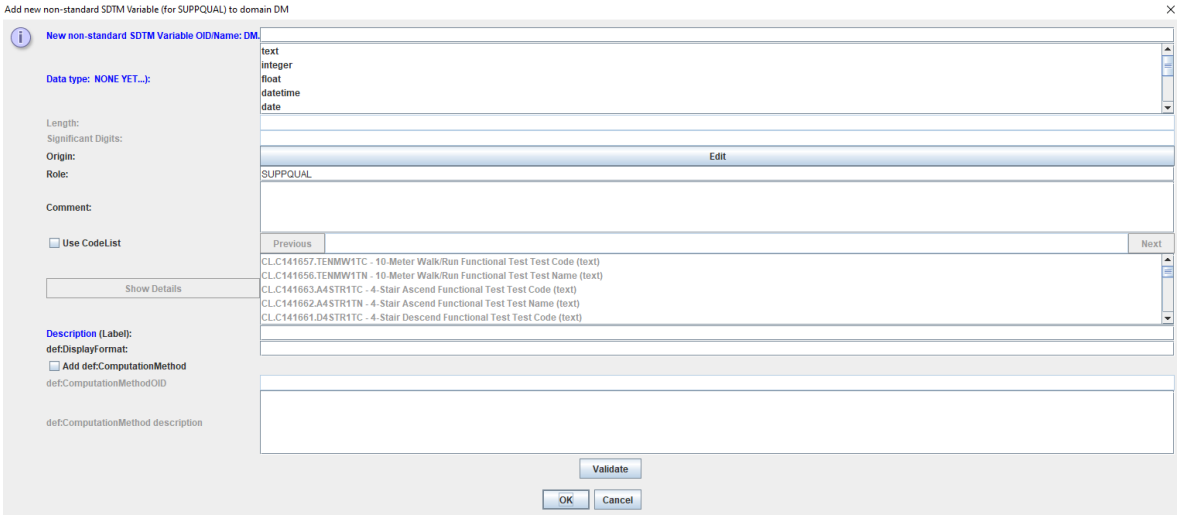
STUDYID	DOMAIN	USUBJID	SUBJID	DM.SITEID	DM.BRTHDTC	DM.SEX
CES	DM	CES-001	001	23	1957-05-07	F
CES	DM	CES-002	002	23	1961-03-04	M
CES	DM	CES-003	003	23	1961-09-06	F
CES	DM	CES-004	004	23	1967-05-01	M
CES	DM	CES-005	005	23	1963-05-21	M
CES	DM	CES-006	006	23	1957-01-01	F
CES	DM	CES-007	007	23	1957-05-31	M
CES	DM	CES-008	008	23	1962-12-31	F
CES	DM	CES-009	009	23	1961-02-27	M
CES	DM	CES-010	010	23	1980-02-01	F

We now want to set up NSVs "RACE1", "RACE2" and "RACE3".

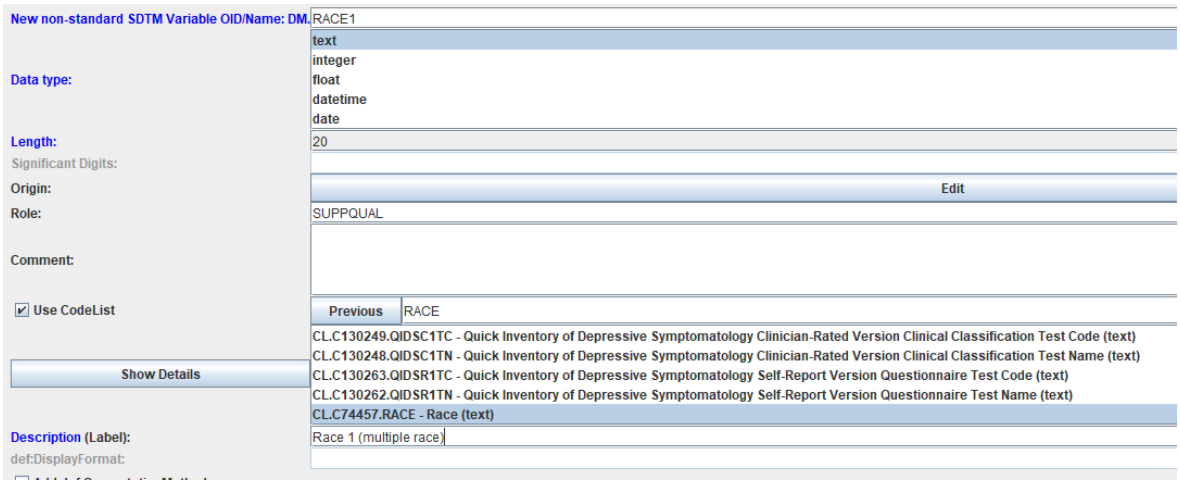
In order to do so, select the DM row, and use the menu "Insert - New non-standard SDTM Variable for SUPPQUAL".



This sets up a wizard:

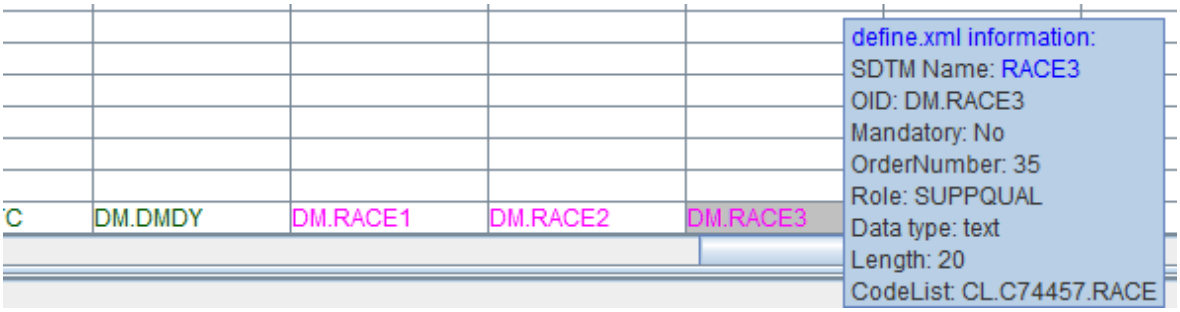


which we fill with the necessary information like:



Giving it the name "RACE1", setting the data type to "text", with a maximal length of 20 (we can later still change this) and assigning the codelist "RACE". For the variable label, we use "Race 1 (multiple race)". This will then later go into SUPPDM.QLABEL.

After clicking "OK" (not a bad idea to use the "Validate" button first), we see that an additional variable has been added to our DM row. It is colored purple, meaning it is an NSV. We then do similar for "RACE2" and "RACE3", with of course a suitable label. This then leads to:



Please ensure that in all 3 cases, you assigned the codelist "RACE" to each of them. If you forgot, or want to change anything, just select the cell, and use the menu "Edit - SDTM

Variable Properties".

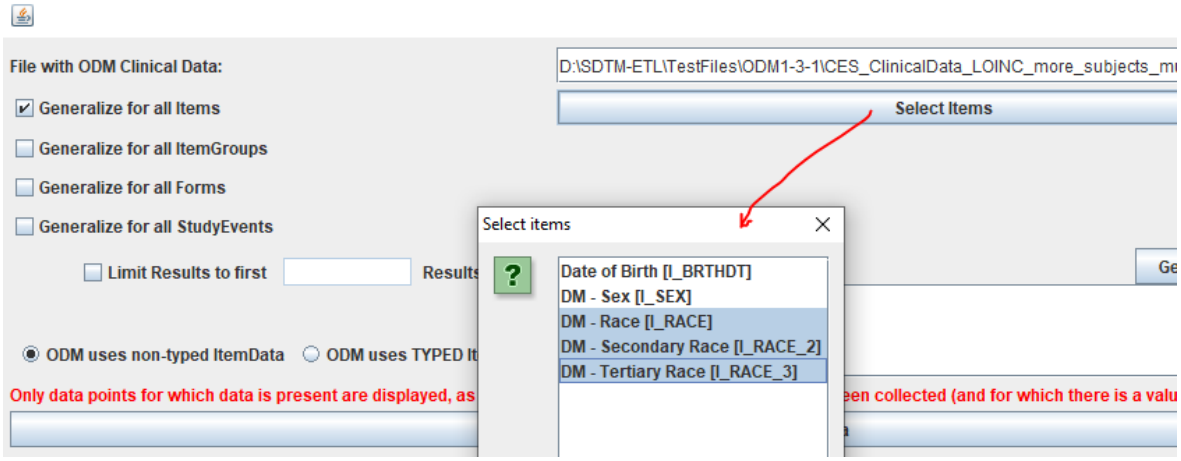
We are now ready to start working on the mapping itself.

Case A: There is a "primary" race

This is the easiest case to map, as the value in DM.RACE is always populated from "Race" in the ODM, independent from whether a secondary and tertiary race are provided.

But first let us look in the ODM clinical data for which subjects there is a secondary and/or tertiary race reported. To do so, select "Race" in the ODM tree, and then use the menu "View - ODM Clinical Data", then check "Generalize for all Items":

and click "Select Items". We then select the three entries for "Race":



After clicking "OK" followed by "View ODM Clinical Data", we then get the table:

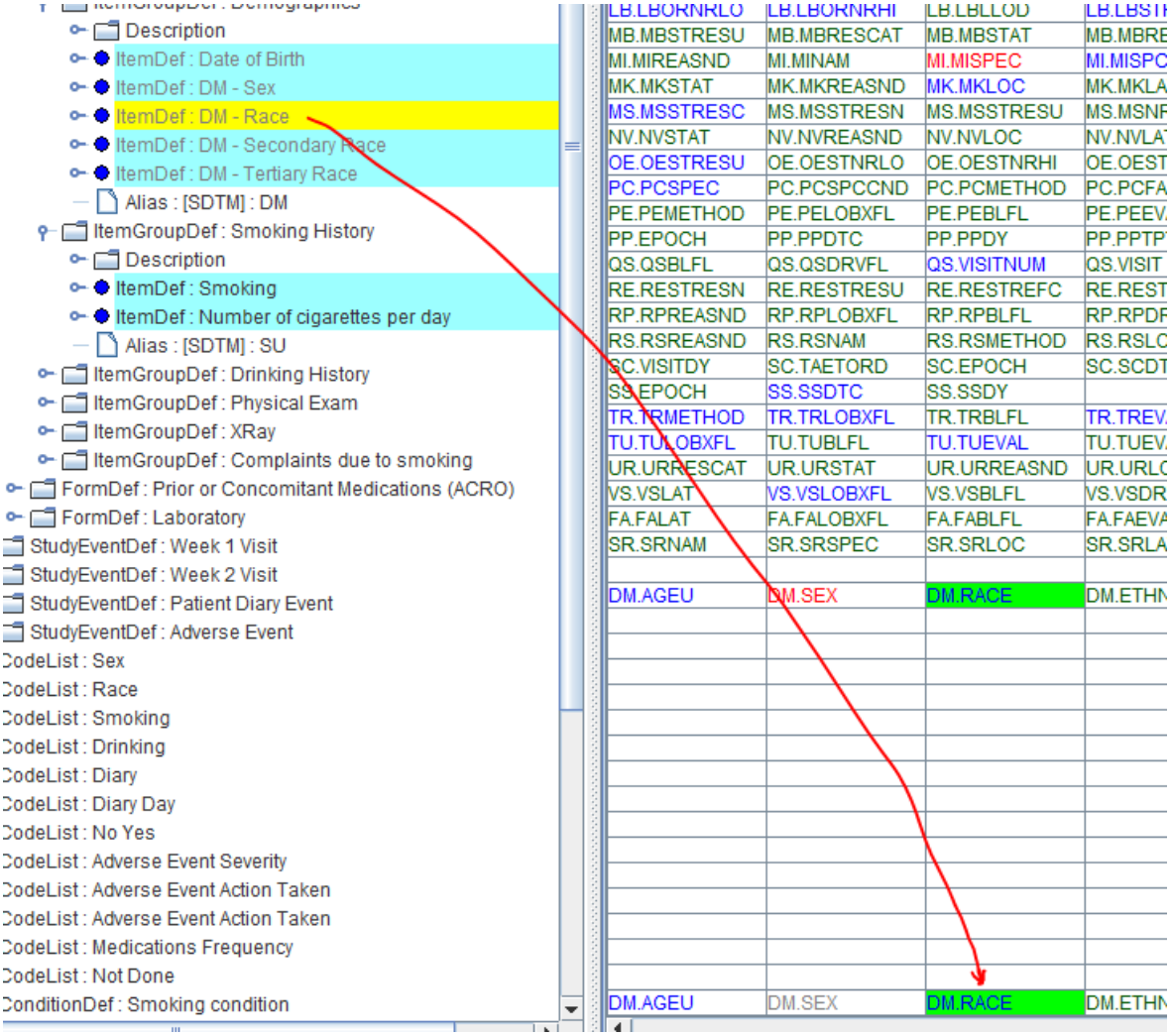
nts for which data is present are displayed, as by default, ODM only contains data that have been collected (and for which there is a value).

View ODM Clinical Data

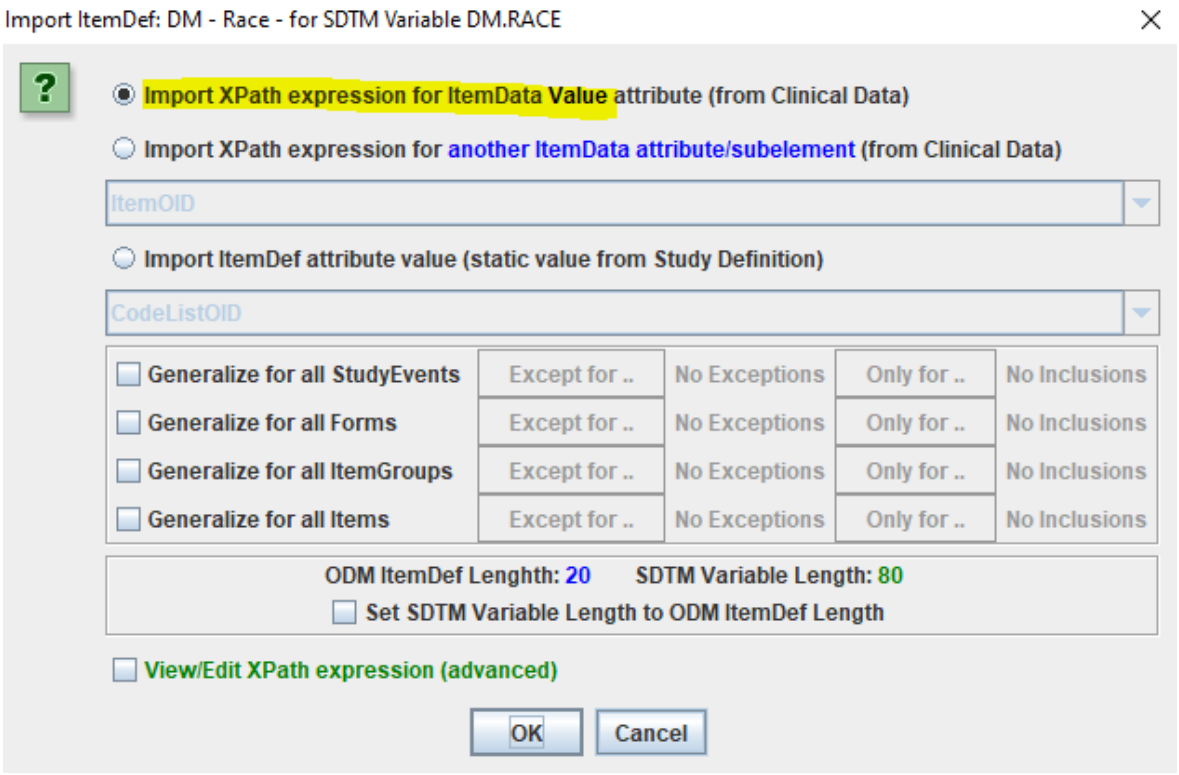
Subject	StudyEvent	Form	ItemGroup	Item	Name	Value
001	BASELINE	F_BASELINE	IG_DM	I_RACE	DM - Race	CAUCASIAN
002	BASELINE	F_BASELINE	IG_DM	I_RACE	DM - Race	ASIAN
003	BASELINE	F_BASELINE	IG_DM	I_RACE	DM - Race	CAUCASIAN
004	BASELINE	F_BASELINE	IG_DM	I_RACE	DM - Race	CAUCASIAN
005	BASELINE	F_BASELINE	IG_DM	I_RACE	DM - Race	CAUCASIAN
006	BASELINE	F_BASELINE	IG_DM	I_RACE	DM - Race	ASIAN
006	BASELINE	F_BASELINE	IG_DM	I_RACE_2	DM - Secondary Race	CAUCASIAN
006	BASELINE	F_BASELINE	IG_DM	I_RACE_3	DM - Tertiary Race	BLACK
007	BASELINE	F_BASELINE	IG_DM	I_RACE	DM - Race	CAUCASIAN
008	BASELINE	F_BASELINE	IG_DM	I_RACE	DM - Race	CAUCASIAN
009	BASELINE	F_BASELINE	IG_DM	I_RACE	DM - Race	CAUCASIAN
010	BASELINE	F_BASELINE	IG_DM	I_RACE	DM - Race	CAUCASIAN

from which we see that only for subject 006, more than one race was reported.

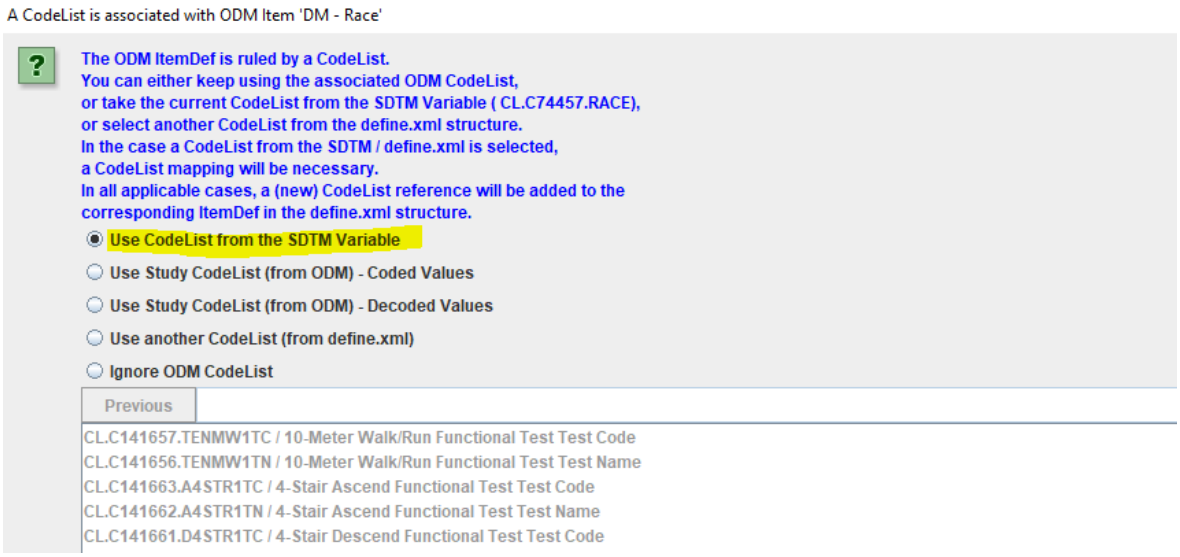
We can now drag-and-drop "Race" to the SDTM cell "DM.RACE":



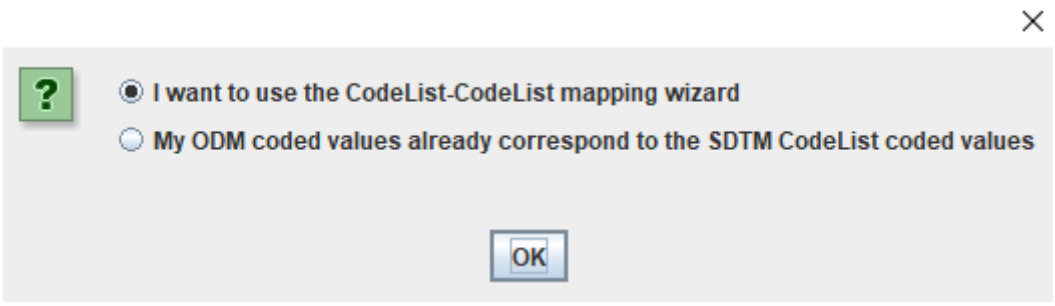
This then triggers the dialog:



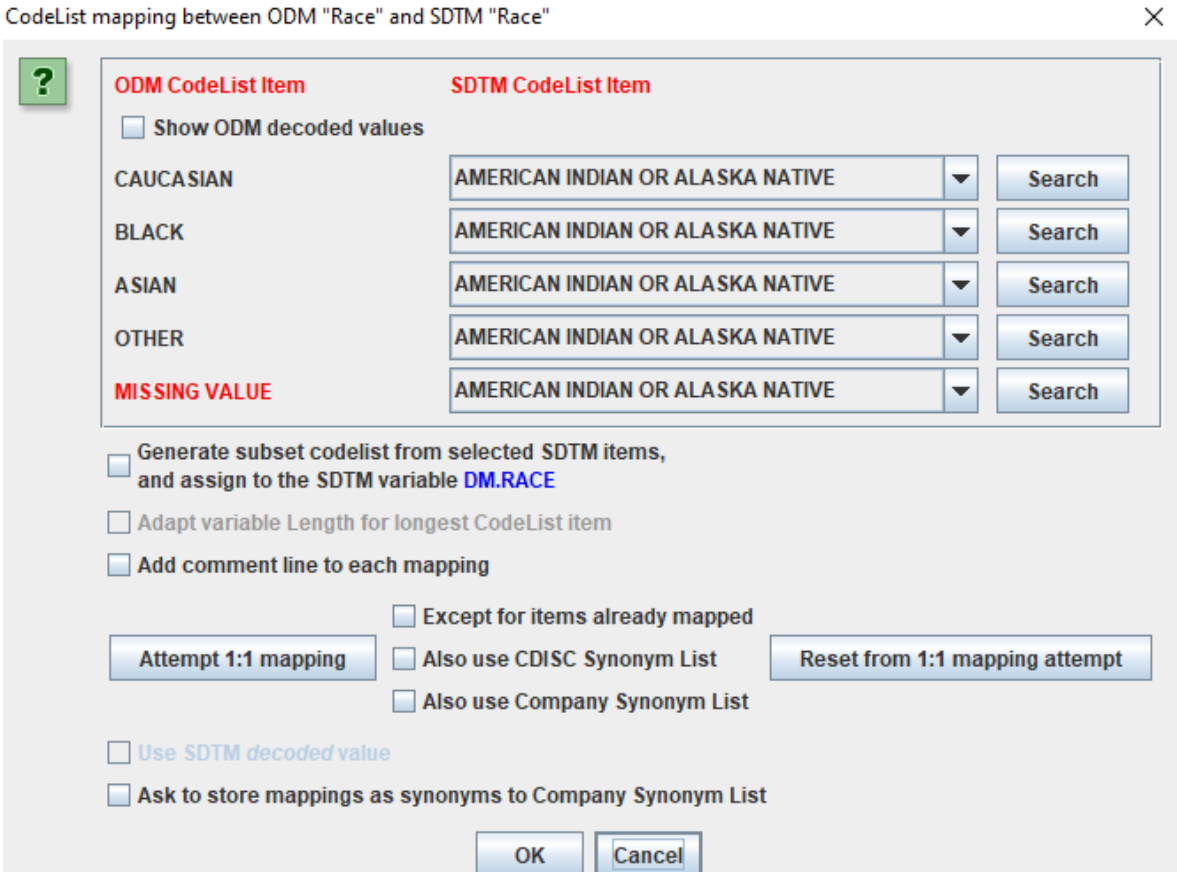
Yes, we want to obtain the collected value, so we let everything as suggested. After clicking "OK", another dialog/wizard is displayed:



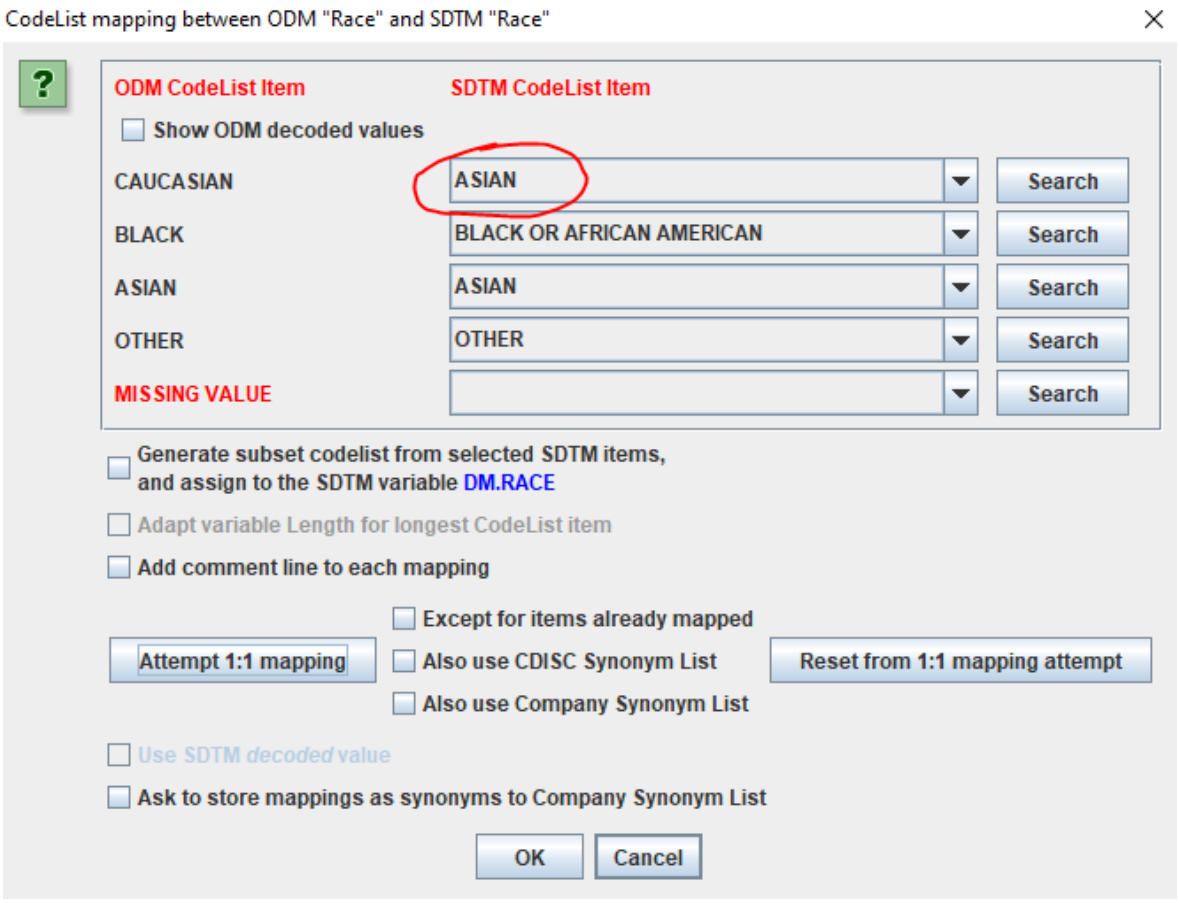
Also here, the system has correctly foreseen that we want to use and map to the SDTM codelist, so we can just click "OK". The following dialog then appears:



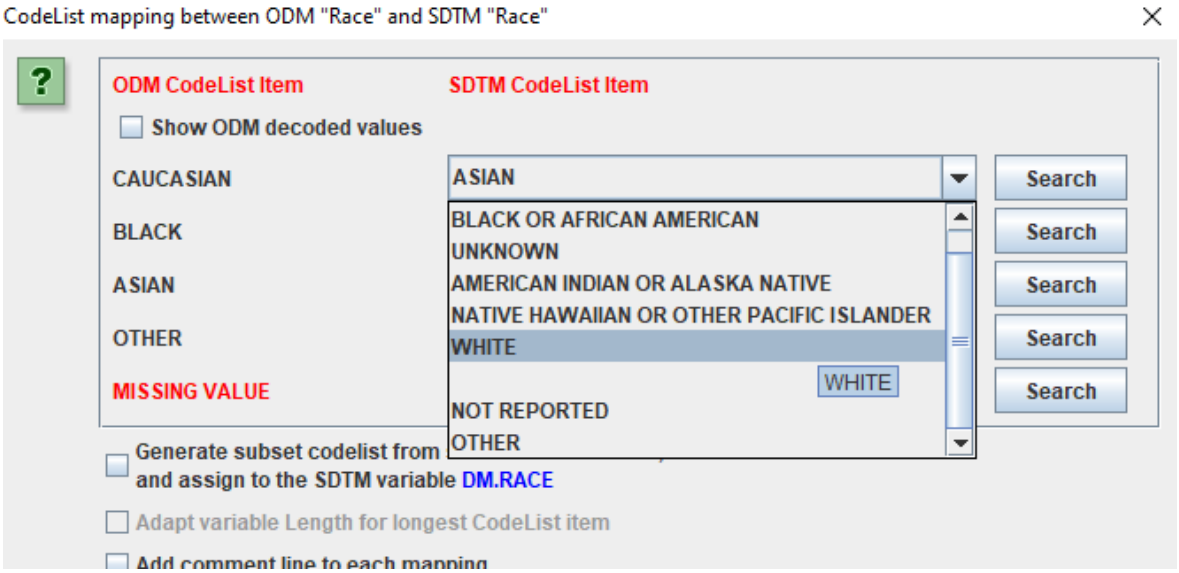
which we can also acknowledge as this is indeed what we want to do. So clicking "OK" leads to:



We can now use the dropboxes to assign the mappings, but we can also be very brave and click the "Attempt 1:1 mappping". This will try to assign the values based on word similarity. The result is:



We see that the generated mapping for "CAUCASIAN" is wrong. This is due to the word similarity between "CAUCASIAN" and "ASIAN". So we need to correct that by using the first dropbox to:



The other ones are just fine, so after clicking "OK", the following mapping script is generated:

The Transformation Script

```
# Mapping using ODM element ItemData with ItemOID I_RACE
# Using SDTM CodeList CL.C74457.RACE
# Using a CodeList mapping between ODM CodeList CL_RACE and SDTM CodeList CL.C74457.RACE
$CODEDVALUE = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData[@FormOID='F_BASELINE']/ItemGroupData[@I
if ($CODEDVALUE == 'CAUCASIAN') {
  $NEWCODEDVALUE = 'WHITE';
} elsif ($CODEDVALUE == 'BLACK') {
  $NEWCODEDVALUE = 'BLACK OR AFRICAN AMERICAN';
} elsif ($CODEDVALUE == 'ASIAN') {
  $NEWCODEDVALUE = 'ASIAN';
} elsif ($CODEDVALUE == 'OTHER') {
  $NEWCODEDVALUE = 'OTHER';
} else {
  $NEWCODEDVALUE = '';
}
$DM.RACE = $NEWCODEDVALUE;

```

looking very good …

Executing the mappings (after clicking "OK"), then ultimately leads to:

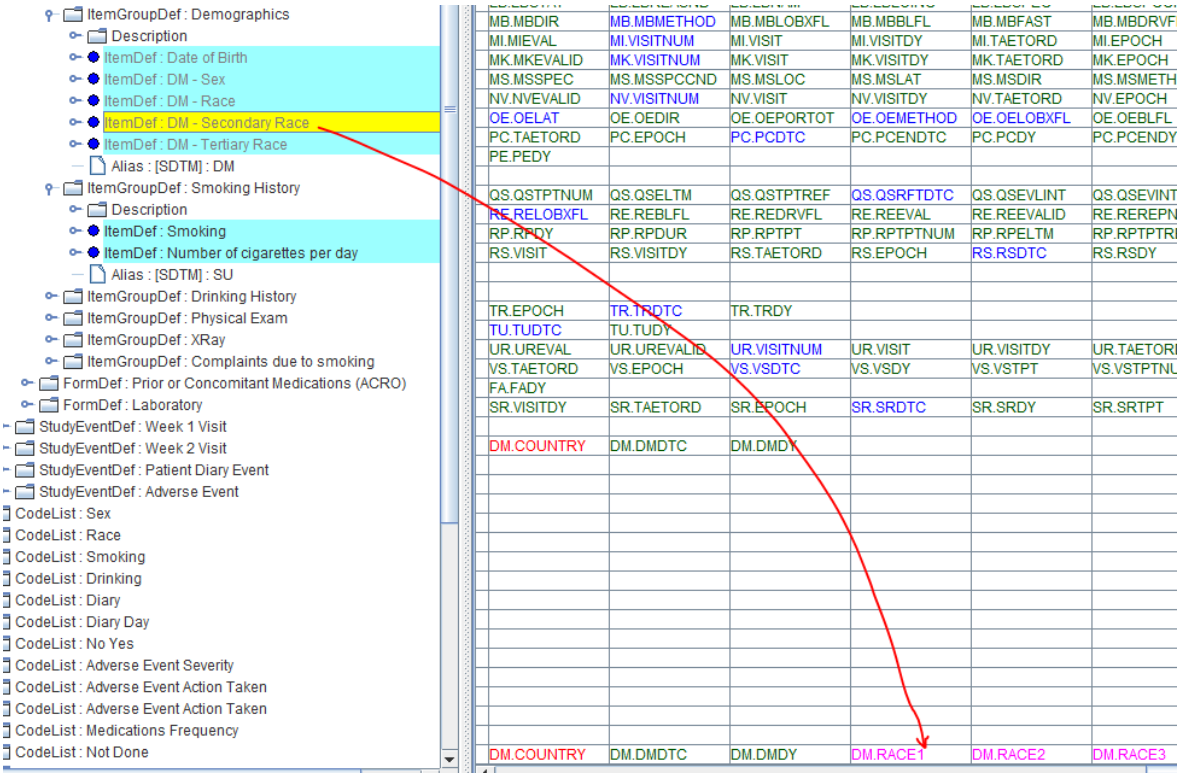
Export selected table as SAS-XPT | Export selected table as Dataset-JSON 1.1 | Export selected table as UTF-8 encoded CSV

CES:DM

STUDYID	DOMAIN	USUBJID	SUBJID	DM.SITEID	DM.BRTHDTC	DM.SEX	DM.RACE
CES	DM	CES-001	001	23	1957-05-07	F	WHITE
CES	DM	CES-002	002	23	1961-03-04	M	ASIAN
CES	DM	CES-003	003	23	1961-09-06	F	WHITE
CES	DM	CES-004	004	23	1967-05-01	M	WHITE
CES	DM	CES-005	005	23	1963-05-21	M	WHITE
CES	DM	CES-006	006	23	1957-01-01	F	ASIAN
CES	DM	CES-007	007	23	1957-05-31	M	WHITE
CES	DM	CES-008	008	23	1962-12-31	F	WHITE
CES	DM	CES-009	009	23	1961-02-27	M	WHITE
CES	DM	CES-010	010	23	1980-02-01	F	WHITE

with "Asian" being assigned as the "primary race" for subject 006. For the other subjects, only 1 value for Race was collected and this is then also appearing (after mapping to the CDISC Controlled Terminology") in the table.

We can now also drag-and-drop "Secondary Race" to the cell "DM.RACE1" as if it were just a normal variable:

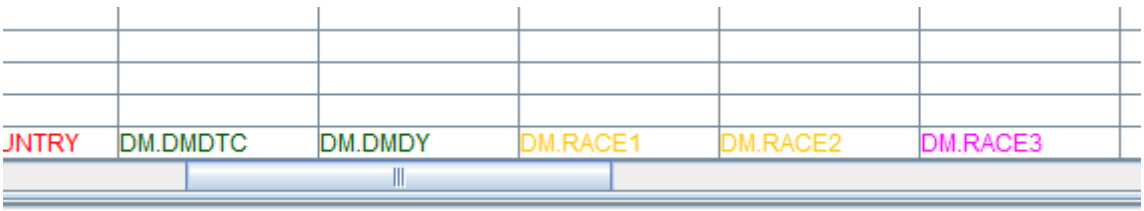


Also here, we then get through all the wizards and dialogs and do the mappings as for "DM.RACE". The finally generated mapping script is then very similar:

The Transformation Script

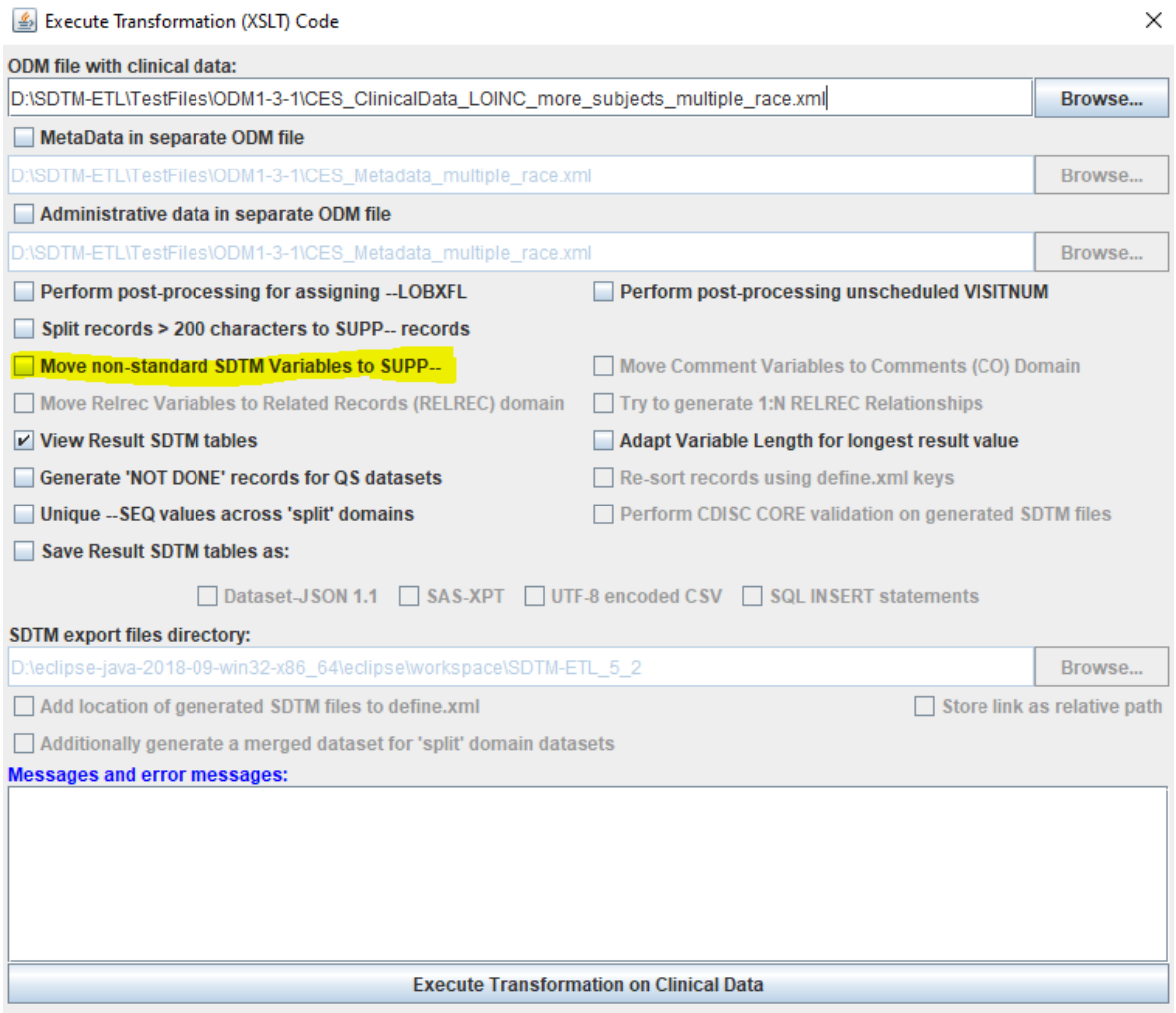
```
1  # Mapping using ODM element ItemData with ItemOID I_RACE_2
2  # Using SDTM CodeList CL.C74457.RACE
3  # Using a CodeList mapping between ODM CodeList CL_RACE and SDTM CodeList CL.C74457.RACE
4  $CODEDVALUE = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData[@FormOID='F_BASELINE']/ItemGroupData[@I
5  if ($CODEDVALUE == 'CAUCASIAN') {
6    $NEWCODEDVALUE = 'WHITE';
7  } elsif ($CODEDVALUE == 'BLACK') {
8    $NEWCODEDVALUE = 'BLACK OR AFRICAN AMERICAN';
9  } elsif ($CODEDVALUE == 'ASIAN') {
10   $NEWCODEDVALUE = 'ASIAN';
11 } elsif ($CODEDVALUE == 'OTHER') {
12   $NEWCODEDVALUE = 'OTHER';
13 } else {
14   $NEWCODEDVALUE = '';
15 }
16 $DM.RACE1 = $NEWCODEDVALUE;
17
18
```

and in the same way, we drag-and-drop "Tertiary Race" to DM.RACE2, as if the latter was were just a normal variable. After having done so, we see that the color of the cells DM.RACE1 and DM.RACE2 has been changed to yellow:

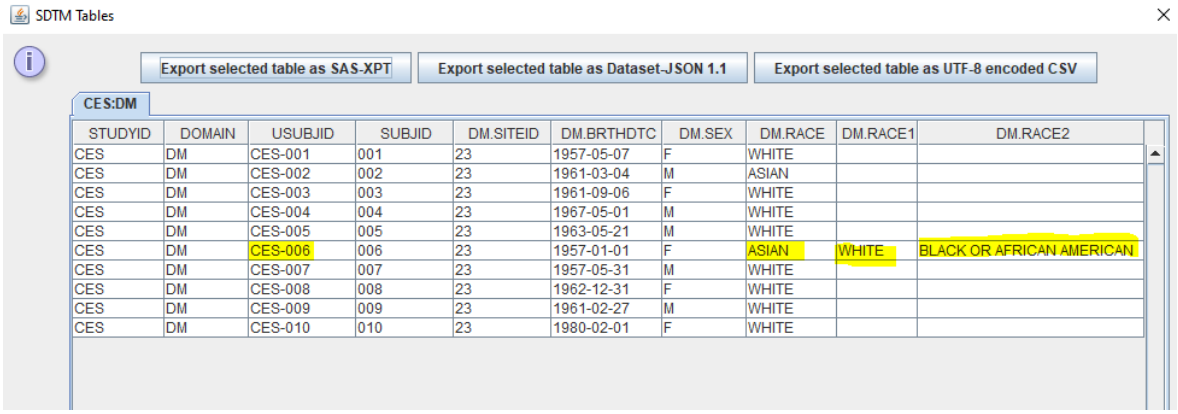


meaning that these cells now contain a mapping.

Let us now test the generated mappings using the menu "Transform - Generate Transformation (XSLT) Code" followed by "Execute Transformation (XSLT) Code", leading to:



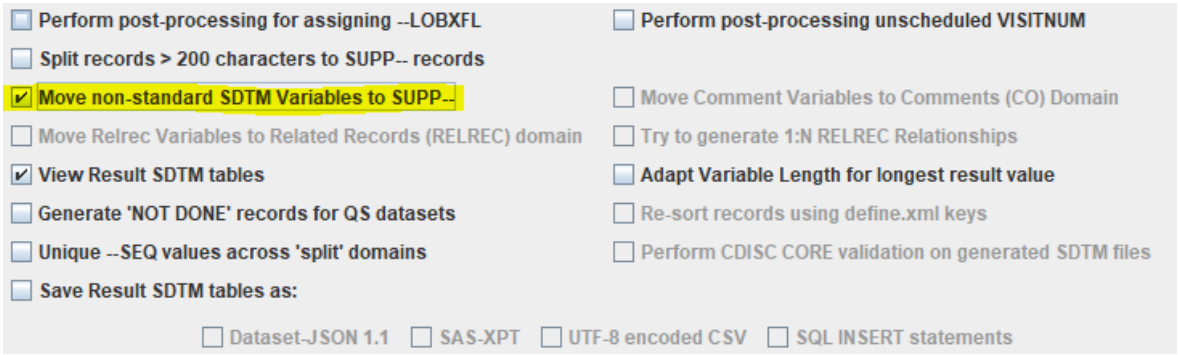
Now comes an important step!
When we are testing, the best is to keep the values for the NSVs "RACE1" and "RACE2" to where they actually belong, which is in the DM dataset!
It is only because the reviewers and their primitive tools do not understand the difference between a "standard" and a "non-standard" variable, that the SDTMIG requires us to "split them off" to SUPPDM. And when the reviewers then want to see them in their original context, they then need to recombine DM with SUPPDM. Looks a bit ridiculous isn't it.
So, if we leave the checkbox "Move non-standard SDTM Variables to SUPP--" UNCHECKED, the result table then becomes:



STUDYID	DOMAIN	USUBJID	SUBJID	DM.SITEID	DM.BRTHDTC	DM.SEX	DM.RACE	DM.RACE1	DM.RACE2
CES	DM	CES-001	001	23	1957-05-07	F	WHITE		
CES	DM	CES-002	002	23	1961-03-04	M	ASIAN		
CES	DM	CES-003	003	23	1961-09-06	F	WHITE		
CES	DM	CES-004	004	23	1967-05-01	M	WHITE		
CES	DM	CES-005	005	23	1963-05-21	M	WHITE		
CES	DM	CES-006	006	23	1957-01-01	F	ASIAN	WHITE	BLACK OR AFRICAN AMERICAN
CES	DM	CES-007	007	23	1957-05-31	M	WHITE		
CES	DM	CES-008	008	23	1962-12-31	F	WHITE		
CES	DM	CES-009	009	23	1961-02-27	M	WHITE		
CES	DM	CES-010	010	23	1980-02-01	F	WHITE		

with the 3 values showing up for subject 006.

When we DO CHECK the checkbox ""Move non-standard SDTM Variables to SUPP--" however:



Then "RACE1" and "RACE2" will be split off, and we get two tables, one for DM, and one for SUPPDM:

CES:DM | CES:SUPPDM

STUDYID	DOMAIN	USUBJID	SUBJID	DM.SITEID	DM.BRTHDTC	DM.SEX	DM.RACE
CES	DM	CES-001	001	23	1957-05-07	F	WHITE
CES	DM	CES-002	002	23	1961-03-04	M	ASIAN
CES	DM	CES-003	003	23	1961-09-06	F	WHITE
CES	DM	CES-004	004	23	1967-05-01	M	WHITE
CES	DM	CES-005	005	23	1963-05-21	M	WHITE
CES	DM	CES-006	006	23	1957-01-01	F	ASIAN
CES	DM	CES-007	007	23	1957-05-31	M	WHITE
CES	DM	CES-008	008	23	1962-12-31	F	WHITE
CES	DM	CES-009	009	23	1961-02-27	M	WHITE
CES	DM	CES-010	010	23	1980-02-01	F	WHITE

and:

CES:DM | CES:SUPPDM

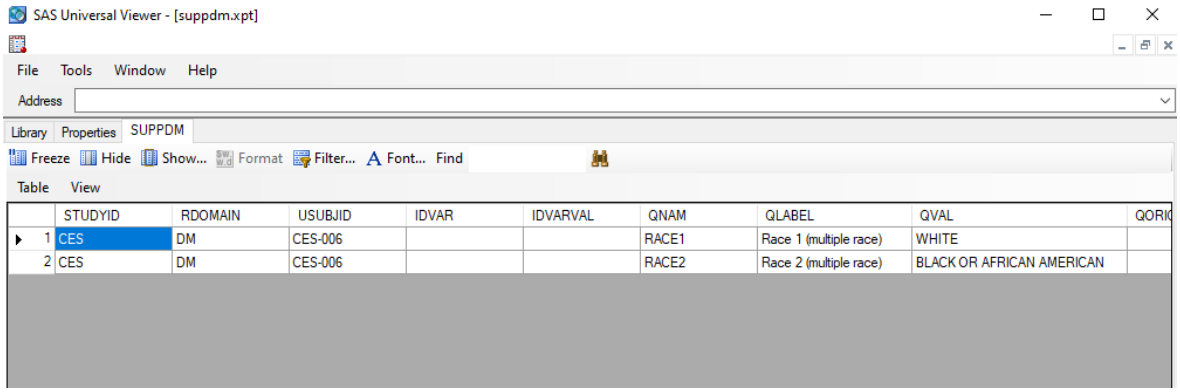
STUDYID	RDOMAIN	USUBJID	IDVAR	IDVARVAL	QNAM	QLABEL	QVAL
CES	DM	CES-006			RACE1	Race 1 (multiple race)	WHITE
CES	DM	CES-006			RACE2	Race 2 (multiple race)	BLACK OR AFRICAN AMERICAN

with two additional records only for subject 006, one for "Race 1 (multiple race"), and "Race 2 (multiple race"). At this point we may decide to the change the label for these. This can

easily be done by going back to the main screen, select e.g. "DM.RACE1" and use the menu "Edit - SDTM Variable Properties", and similar for "DM.RACE2".

When we are nearing final mappings (we e.g. haven't done anything on "DM.AGE" yet), we can also generate the SAS-XPT files by checking "Save Result SDTM tables as" and "SAS-XPT". At this moment, we can then also ask the system to adapt the variable lengths to the length of the longest real value by checking the checkbox "Adapt Variable Length for longest result value", in order to keep file size as small as possible[1].

The result SUPPDM-XPT file then looks like:



	STUDYID	RDOMAIN	USUBJID	IDVAR	IDVARVAL	QNAM	QLABEL	QVAL	QORIG
1	CES	DM	CES-006			RACE1	Race 1 (multiple race)	WHITE	
2	CES	DM	CES-006			RACE2	Race 2 (multiple race)	BLACK OR AFRICAN AMERICAN	

Remark that for SUPPDM, IDVAR and IDVARVAL will never be populated, as there is only 1 record per subject in the "parent" DM dataset, so there is nothing like "DMSEQ".
In case of other domains, the system will automatically fill IDVAR with the "Sequence Number" variable, e.g. LBSEQ in the case of SUPPLB, and calculate the correct value of it and put that in IDVARVAL.

Case B: Collected races are equivalent

In this case, we need to populate DM.RACE with "MULTIPLE" when more than one race values was reported. This makes the mapping slightly more complicated (but it's still "easy of you know how"), as we need to check whether a second and third race have been reported.

Again, we start by a drag-and-drop of "Race" to the SDTM cell "DM.RACE", and go through the wizards helping us to map from our "local" race to the SDTM one. The generated script then is:

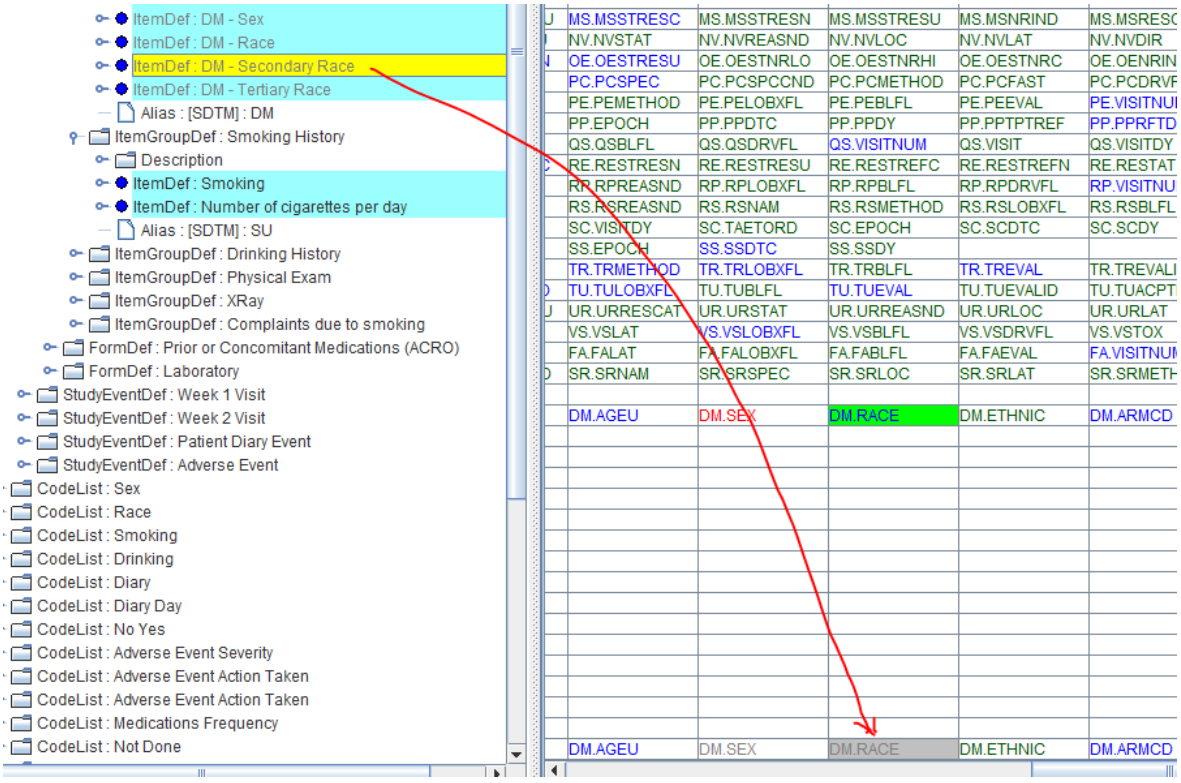
[1] Essentially for DM and SUPPDM, this is ridiculous, as these files never get large. but it avoids that e.g. starts to complain about it (it shouldn't) which means we would need to write an explanation in the "Reviewers Guide". This is one of the reasons also FDA should move to CDISC CORE, and forget about P21.

The Transformation Script

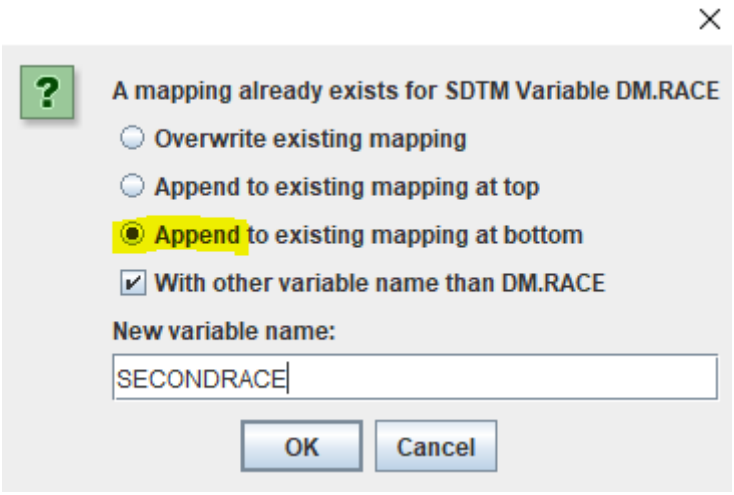
```
# Mapping using ODM element ItemData with ItemOID I_RACE
# Using SDTM CodeList CL.C74457.RACE
# Using a CodeList mapping between ODM CodeList CL_RACE and SDTM CodeList CL.C74457.RACE
$CODEDVALUE = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData[@FormOID='F_BASELINE']/ItemGrou
if ($CODEDVALUE == 'CAUCASIAN') {
  $NEWCODEDVALUE = 'WHITE';
} elsif ($CODEDVALUE == 'BLACK') {
  $NEWCODEDVALUE = 'BLACK OR AFRICAN AMERICAN';
} elsif ($CODEDVALUE == 'ASIAN') {
  $NEWCODEDVALUE = 'ASIAN';
} elsif ($CODEDVALUE == 'OTHER') {
  $NEWCODEDVALUE = 'OTHER';
} else {
  $NEWCODEDVALUE = '';
}
$DM.RACE = $NEWCODEDVALUE;


```

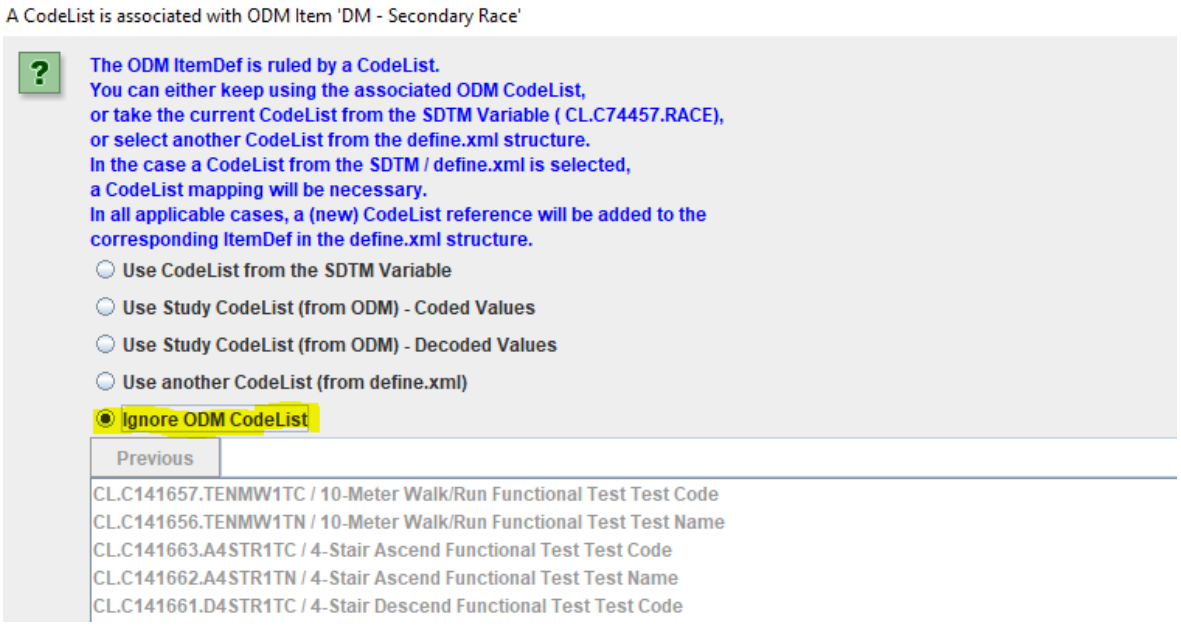
This is however not sufficient as we also need to know whether a second and/or third race has been provided. In order to know this, drag-and-drop "Secondary race" (we will treat it is as if it was not "secondary" just for the exercise) to the SAME DM.RACE cell in the right part:



and "append" it to the existing script with another name, like "SECONDRACE":



As we just want to know whether there is one, we do NOT map to the SDTM values, but just ignore that there is a codelist for it at the ODM side:



and after clicking "OK", leading to the extended script:

The Transformation Script

```
1  # Mapping using ODM element ItemData with ItemOID I_RACE
2  # Using SDTM CodeList CL.C74457.RACE
3  # Using a CodeList mapping between ODM CodeList CL_RACE and SDTM CodeList CL.C74457.RACE
4  $CODEDVALUE = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData[@FormOID='F_BASELINE']/ItemGrou
5  if ($CODEDVALUE == 'CAUCASIAN') {
6    $NEWCODEDVALUE = 'WHITE';
7  } elsif ($CODEDVALUE == 'BLACK') {
8    $NEWCODEDVALUE = 'BLACK OR AFRICAN AMERICAN';
9  } elsif ($CODEDVALUE == 'ASIAN') {
10   $NEWCODEDVALUE = 'ASIAN';
11 } elsif ($CODEDVALUE == 'OTHER') {
12   $NEWCODEDVALUE = 'OTHER';
13 } else {
14   $NEWCODEDVALUE = '';
15 }
16 $DM.RACE = $NEWCODEDVALUE;
17 # Mapping using ODM element ItemData with ItemOID I_RACE_2
18 # Ignoring the ODM CodeList
19 $SECONDRACE = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData[@FormOID='F_BASELINE']/ItemGrou
20
```

We now do the same for "Tertiary Race", and, on the SDTM side, name it "THIRDRACE", leading to:

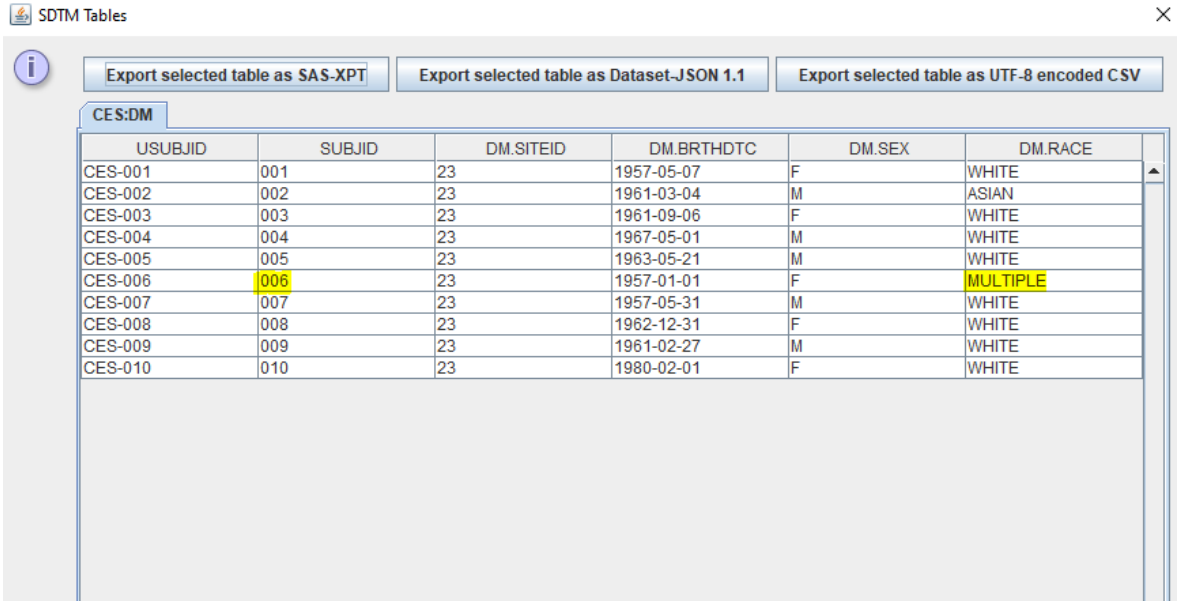
```
11 } elsif ($CODEDVALUE == 'OTHER') {
12   $NEWCODEDVALUE = 'OTHER';
13 } else {
14   $NEWCODEDVALUE = '';
15 }
16 $DM.RACE = $NEWCODEDVALUE;
17 # Mapping using ODM element ItemData with ItemOID I_RACE_2
18 # Ignoring the ODM CodeList
19 $SECONDRACE = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData[@FormOID='F_BASELI
20 # Mapping using ODM element ItemData with ItemOID I_RACE_3
21 # Ignoring the ODM CodeList
22 $THIRDRACE = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData[@FormOID='F_BASELIN
23
```

Now we do see that the statement on line 16 cannot be correct, as it doesn't take the check on second and third race into account. So we delete line 16, and start an if-else structure at the bottom, checking whether there is a value for the second and third race: if there is, we assign "MULTIPLE" to DM.RACE, if not, we just assign the value of the first race:

```
16 # Mapping using ODM element ItemData with ItemOID I_RACE_2
17 # Ignoring the ODM CodeList
18 $SECONDRACE = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData[@FormOID='F_BASELINE']/Ite
19 # Mapping using ODM element ItemData with ItemOID I_RACE_3
20 # Ignoring the ODM CodeList
21 $THIRDRACE = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData[@FormOID='F_BASELINE']/Item
22 # checking whether there is a value for the second and third race:
23 # if there is, we assign "MULTIPLE" to DM.RACE,
24 # if not, we just assign the value of the first race
25 if($SECONDRACE != '' or $THIRDRACE != '') {
26   $DM.RACE = 'MULTIPLE';
27 }  else {
28   $DM.RACE = $NEWCODEDVALUE;
29 }
```

In line 25, we check on whether the second or third race is not empty. If this is the case, we assign "MULTIPLE".

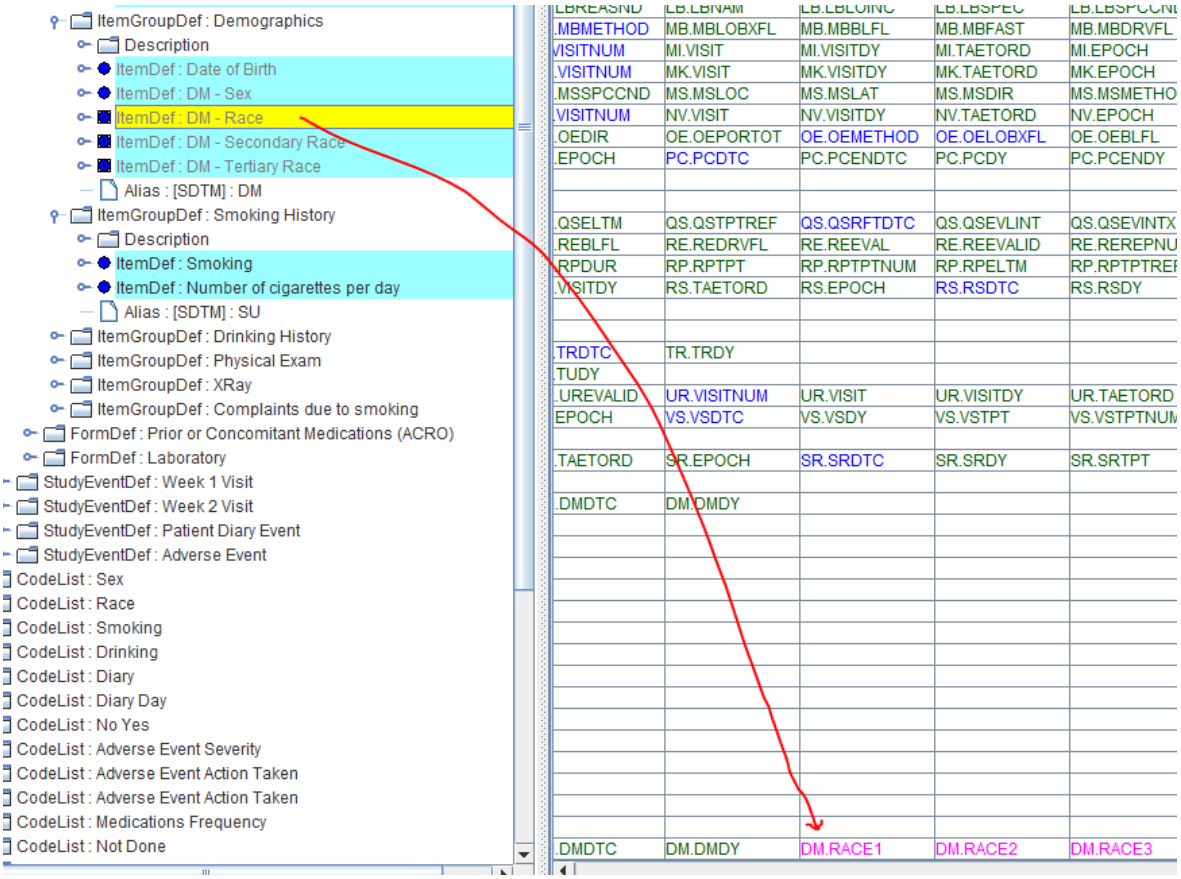
Then executing the script using the menu "Transform" … the result is:



USUBJID	SUBJID	DM.SITEID	DM.BRTHDTC	DM.SEX	DM.RACE
CES-001	001	23	1957-05-07	F	WHITE
CES-002	002	23	1961-03-04	M	ASIAN
CES-003	003	23	1961-09-06	F	WHITE
CES-004	004	23	1967-05-01	M	WHITE
CES-005	005	23	1963-05-21	M	WHITE
CES-006	006	23	1957-01-01	F	MULTIPLE
CES-007	007	23	1957-05-31	M	WHITE
CES-008	008	23	1962-12-31	F	WHITE
CES-009	009	23	1961-02-27	M	WHITE
CES-010	010	23	1980-02-01	F	WHITE

with the value "MULTIPLE" for subject 006.

For the NSVs "RACE1", "RACE2", and "RACE3", we need the ODM values from "Race", "Secondary Race" and "Tertiary Race", but only when the assigned value for SDTM DM.RACE is not "MULTIPLE".
So we drag-and-drop "Race" on the ODM side to the cell DM.RACE1, which is the cell for the NSV:



and do the mapping to the SDTM codelist, leading to:

The Transformation Script

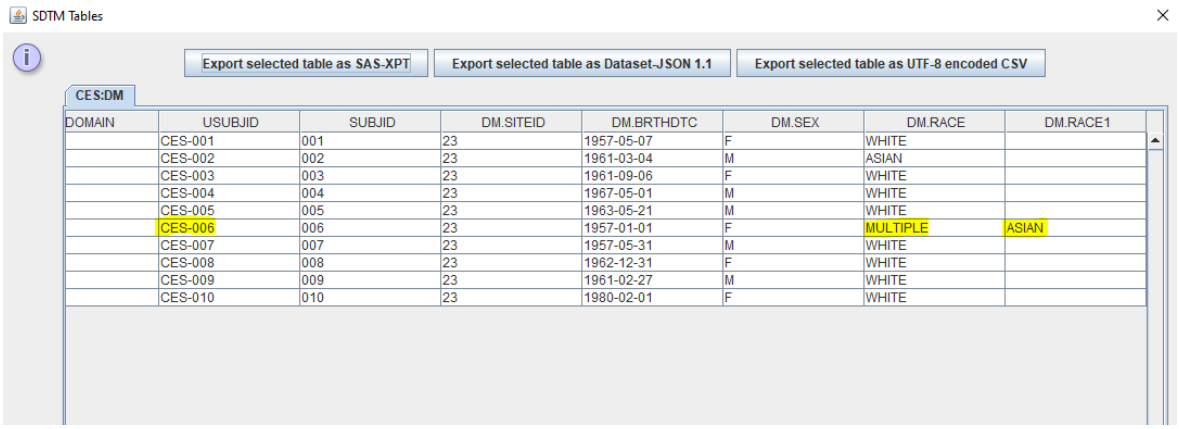
```
1  # Mapping using ODM element ItemData with ItemOID I_RACE
2  # Using SDTM CodeList CL.C74457.RACE
3  # Using a CodeList mapping between ODM CodeList CL_RACE and SDTM CodeList CL.C74457.RACE
4  $CODEDVALUE = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData[@FormOID='F_BASELINE']/ItemGr
5  if ($CODEDVALUE == 'CAUCASIAN') {
6    $NEWCODEDVALUE = 'WHITE';
7  } elsif ($CODEDVALUE == 'BLACK') {
8    $NEWCODEDVALUE = 'BLACK OR AFRICAN AMERICAN';
9  } elsif ($CODEDVALUE == 'ASIAN') {
10   $NEWCODEDVALUE = 'ASIAN';
11 } elsif ($CODEDVALUE == 'OTHER') {
12   $NEWCODEDVALUE = 'OTHER';
13 } else {
14   $NEWCODEDVALUE = '';
15 }
16 $DM.RACE1 = $NEWCODEDVALUE;
17
18
```

If we now would let it "as is", the NSV "RACE1" would always be populated with the first race reported, but it should only be populated in the case that DM.RACE is "MULTIPLE".

Now, in SDTM-ETL, we can always ask up the value of another variable that has been defined before, i.e. from a cell to the left of the current one. As "DM.RACE" is left from "DM.RACE1", we can ask for the value of $DM.RACE in the script of DM.RACE1, and test on it. This leads to the script (in DM.RACE1):

```
} elsif ($CODEDVALUE == 'ASIAN') {
  $NEWCODEDVALUE = 'ASIAN';
} elsif ($CODEDVALUE == 'OTHER') {
  $NEWCODEDVALUE = 'OTHER';
} else {
  $NEWCODEDVALUE = '';
}
# only allow RACE1 to be populated when the value of RACE (the standard variable) is 'MULTIPLE'
# When not, leave empty
if($DM.RACE == 'MULTIPLE') {
      $DM.RACE1 = $NEWCODEDVALUE;
} else {
      $DM.RACE1 = '';
}
```

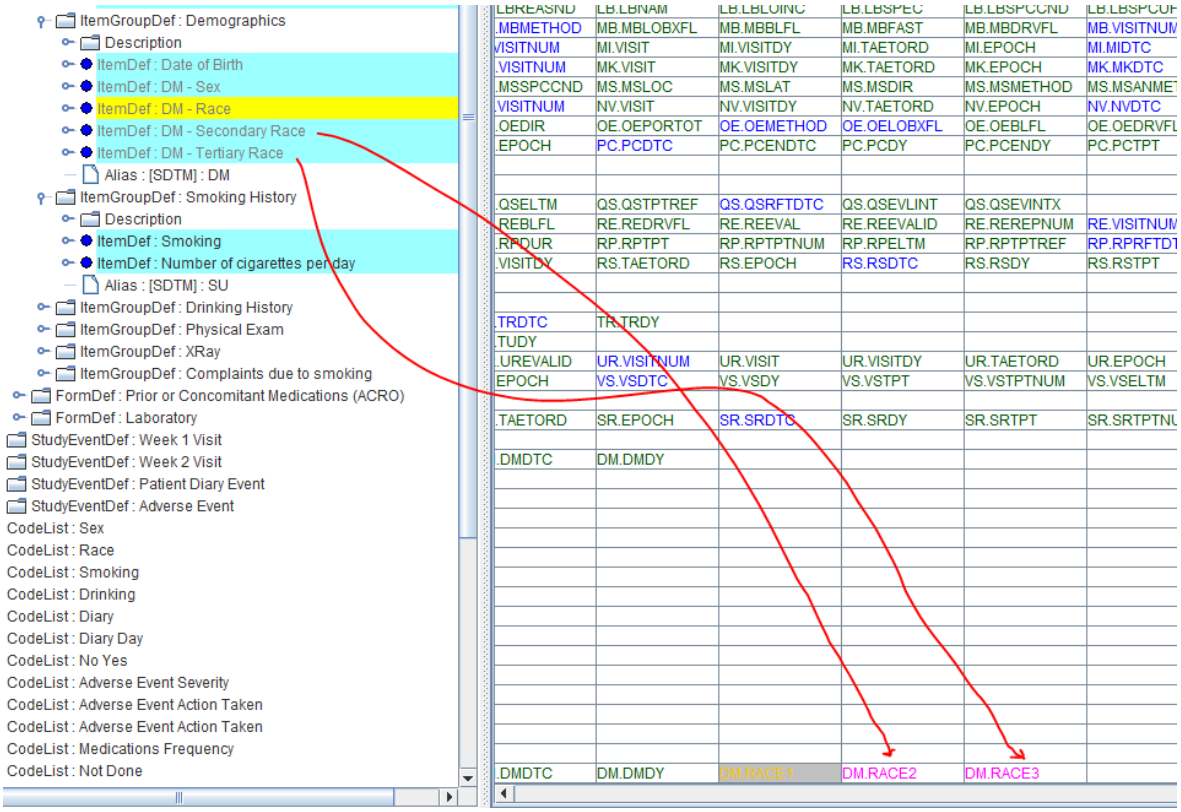
When executing the mappings, and leaving the checkbox "Move non-standard SDTM Variables to SUPPQUAL" UNCHECKED", leading to the result:



DOMAIN	USUBJID	SUBJID	DM.SITEID	DM.BRTHDTC	DM.SEX	DM.RACE	DM.RACE1
	CES-001	001	23	1957-05-07	F	WHITE	
	CES-002	002	23	1961-03-04	M	ASIAN	
	CES-003	003	23	1961-09-06	F	WHITE	
	CES-004	004	23	1967-05-01	M	WHITE	
	CES-005	005	23	1963-05-21	M	WHITE	
	CES-006	006	23	1957-01-01	F	MULTIPLE	ASIAN
	CES-007	007	23	1957-05-31	M	WHITE	
	CES-008	008	23	1962-12-31	F	WHITE	
	CES-009	009	23	1961-02-27	M	WHITE	
	CES-010	010	23	1980-02-01	F	WHITE	

with, for subject 006, the value for DM.RACE being "MULTIPLE", and the first race, "ASIAN" going into the NSV "DM.RACE1".

We can now do similar for the "second race" (drag-and-drop to DM.RACE2) and the "third race" (to DM.RACE3), taking care again of the SDTM codelist:

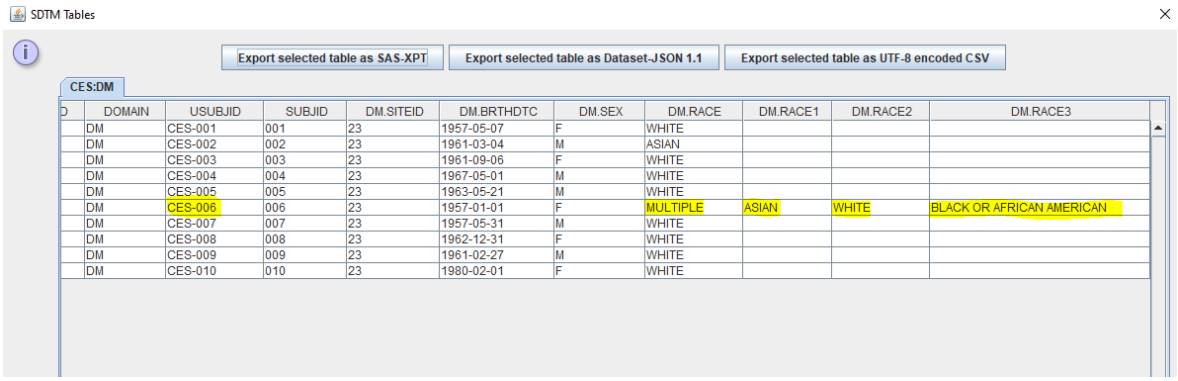


We can again check on "MULTIPLE" in DM.RACE, but this is essentially not necessary, as there cannot be a second or third race when DM.RACE is not "MULTIPLE".
The mapping script for the NSV "DM.RACE2" then becomes:

The Transformation Script

```
1  # Mapping using ODM element ItemData with ItemOID I_RACE_2
2  # Using SDTM CodeList CL.C74457.RACE
3  # Using a CodeList mapping between ODM CodeList CL_RACE and SDTM CodeList CL.C74457.RAC
4  $CODEDVALUE = xpath(/StudyEventData[@StudyEventOID='BASELINE']/FormData[@FormOID='F_BAS
5  if ($CODEDVALUE == 'CAUCASIAN') {
6    $NEWCODEDVALUE = 'WHITE';
7  } elsif ($CODEDVALUE == 'BLACK') {
8    $NEWCODEDVALUE = 'BLACK OR AFRICAN AMERICAN';
9  } elsif ($CODEDVALUE == 'ASIAN') {
10   $NEWCODEDVALUE = 'ASIAN';
11 } elsif ($CODEDVALUE == 'OTHER') {
12   $NEWCODEDVALUE = 'OTHER';
13 } else {
14   $NEWCODEDVALUE = '';
15 }
16 $DM.RACE2 = $NEWCODEDVALUE;
17
```

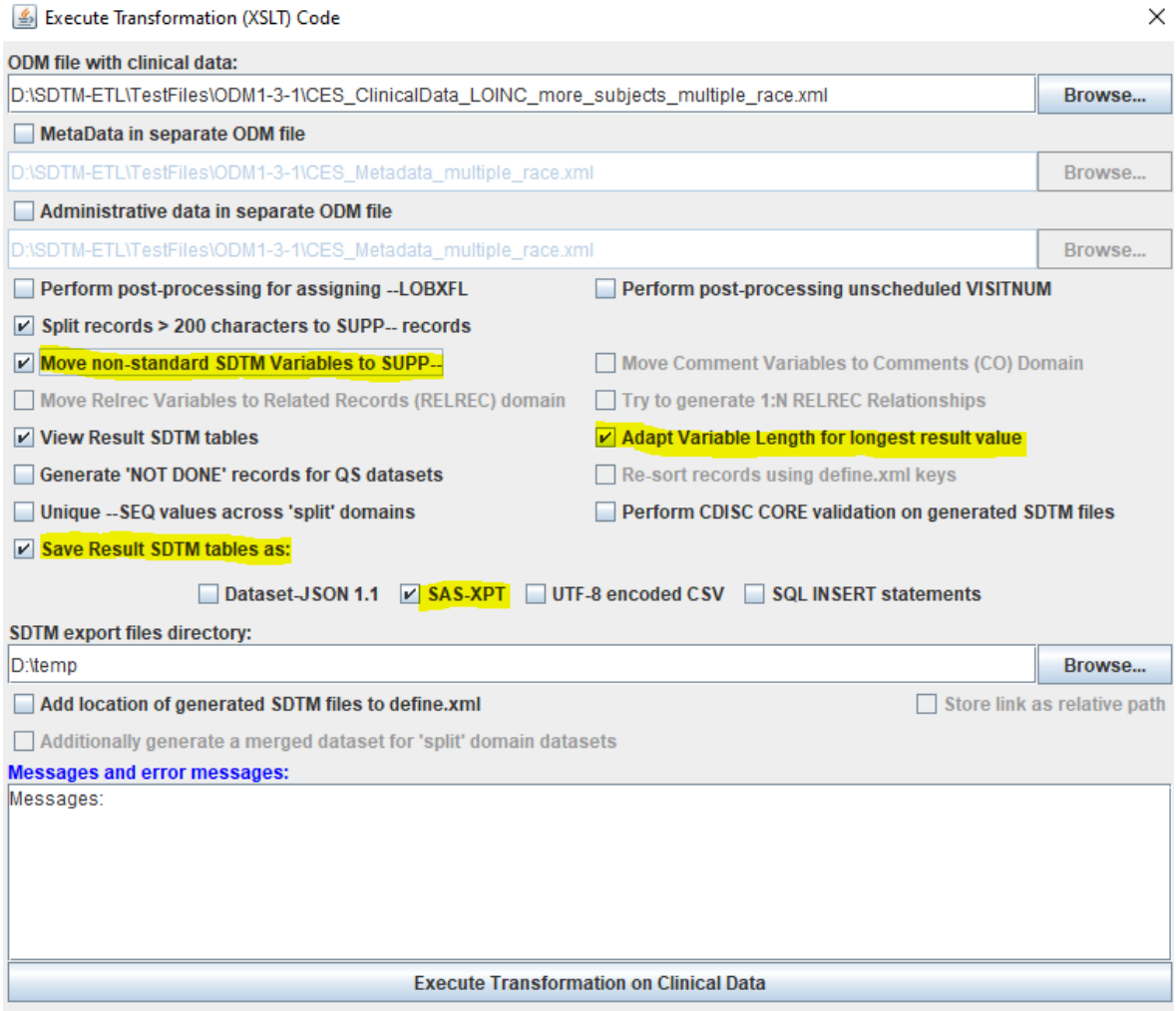
Execution of our mapping scripts then leads to:



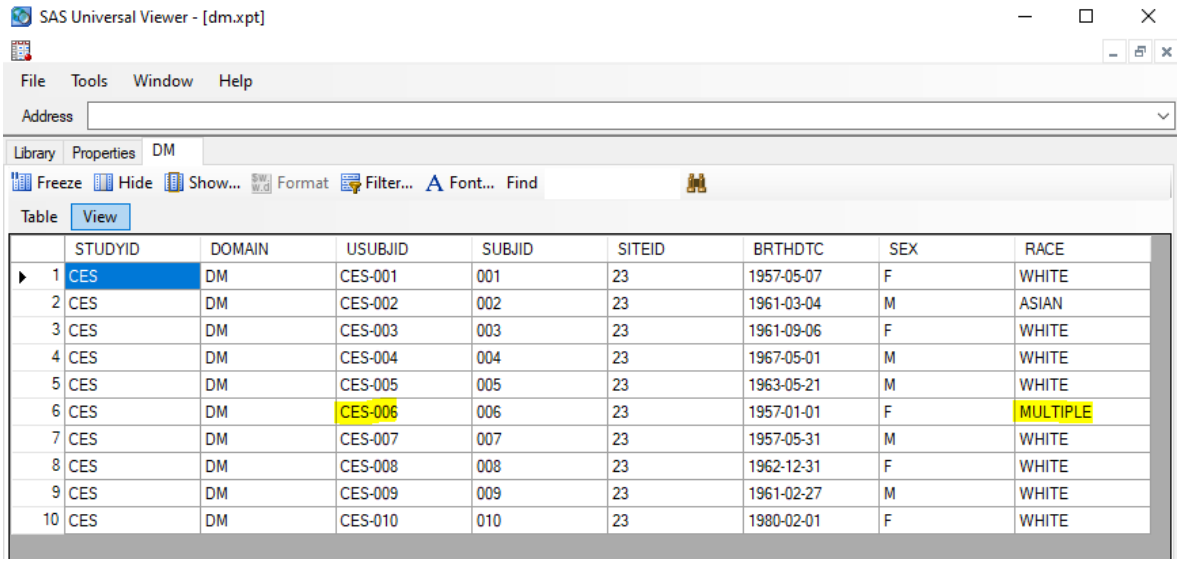
	DOMAIN	USUBJID	SUBJID	DM.SITEID	DM.BRTHDTC	DM.SEX	DM.RACE	DM.RACE1	DM.RACE2	DM.RACE3
	DM	CES-001	001	23	1957-05-07	F	WHITE			
	DM	CES-002	002	23	1961-03-04	M	ASIAN			
	DM	CES-003	003	23	1961-09-06	F	WHITE			
	DM	CES-004	004	23	1967-05-01	M	WHITE			
	DM	CES-005	005	23	1963-05-21	M	WHITE			
	DM	CES-006	006	23	1957-01-01	F	MULTIPLE	ASIAN	WHITE	BLACK OR AFRICAN AMERICAN
	DM	CES-007	007	23	1957-05-31	M	WHITE			
	DM	CES-008	008	23	1962-12-31	F	WHITE			
	DM	CES-009	009	23	1961-02-27	M	WHITE			
	DM	CES-010	010	23	1980-02-01	F	WHITE			

which is exactly what we want.

When we then want to generate SAS-XPT files and move "RACE1", "RACE2" and "RACE3" to SUPPDM, we need to check the checkbox "Move non-standard SDTM Variables to SUPPQUAL". In order to "optimize" the file size for both DM and SUPPDM, we also check the checkbox "Adapt Variable Length for longest result value":

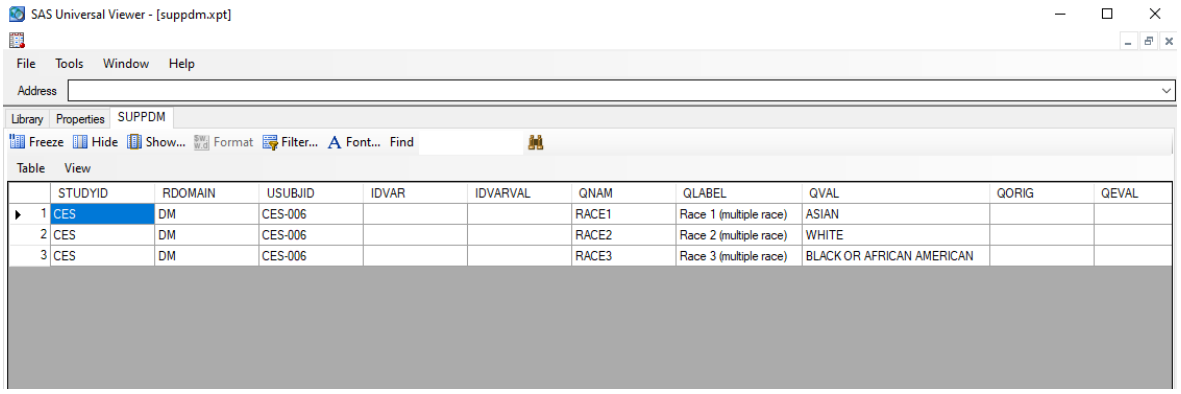


When then executing, two SAS-XPT files are generated, "dm.xpt" and "suppdm.xpt". The content of the dm.xpt file is:



	STUDYID	DOMAIN	USUBJID	SUBJID	SITEID	BRTHDTC	SEX	RACE
1	CES	DM	CES-001	001	23	1957-05-07	F	WHITE
2	CES	DM	CES-002	002	23	1961-03-04	M	ASIAN
3	CES	DM	CES-003	003	23	1961-09-06	F	WHITE
4	CES	DM	CES-004	004	23	1967-05-01	M	WHITE
5	CES	DM	CES-005	005	23	1963-05-21	M	WHITE
6	CES	DM	CES-006	006	23	1957-01-01	F	MULTIPLE
7	CES	DM	CES-007	007	23	1957-05-31	M	WHITE
8	CES	DM	CES-008	008	23	1962-12-31	F	WHITE
9	CES	DM	CES-009	009	23	1961-02-27	M	WHITE
10	CES	DM	CES-010	010	23	1980-02-01	F	WHITE

and the content of suppdm.xpt is:



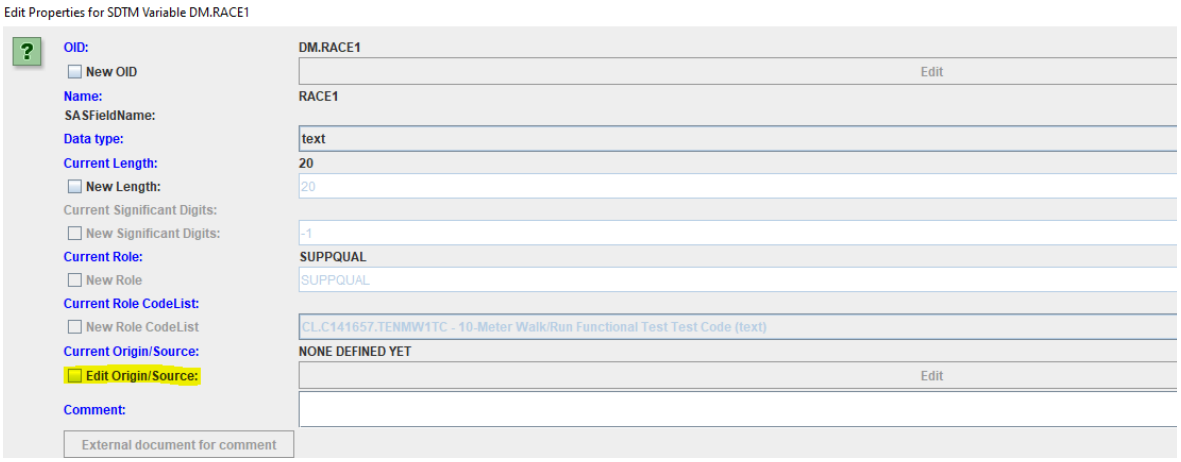
	STUDYID	RDOMAIN	USUBJID	IDVAR	IDVARVAL	QNAM	QLABEL	QVAL	QORIG	QEVAL
1	CES	DM	CES-006			RACE1	Race 1 (multiple race)	ASIAN		
2	CES	DM	CES-006			RACE2	Race 2 (multiple race)	WHITE		
3	CES	DM	CES-006			RACE3	Race 3 (multiple race)	BLACK OR AFRICAN AMERICAN		

with 3 rows for subject 006, one for RACE1, one for RACE2, and one for RACE3.

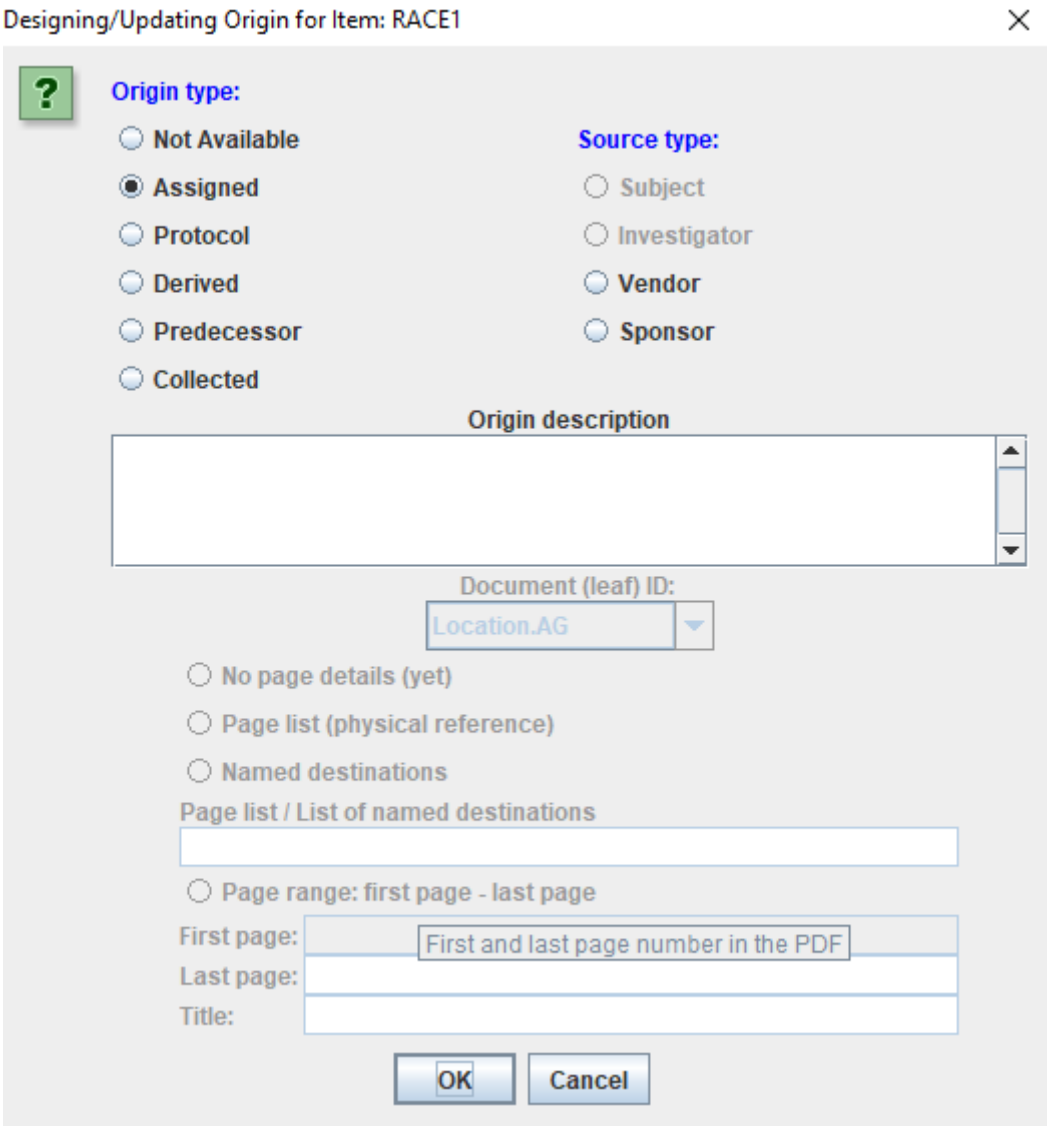
Adding a value for QORIG in SUPPDM

If one checks the SDTMIG, one sees that in any SUPPxx dataset, QORIG must be populated. Essentially this is idiotic, as it is metadata, and should thus not be in SDTM, and the "origin" is provided in the define.xml anyway. This requirement still stems from the time that the SDTM development team did not understand the Define-XML standard, and has never been corrected.

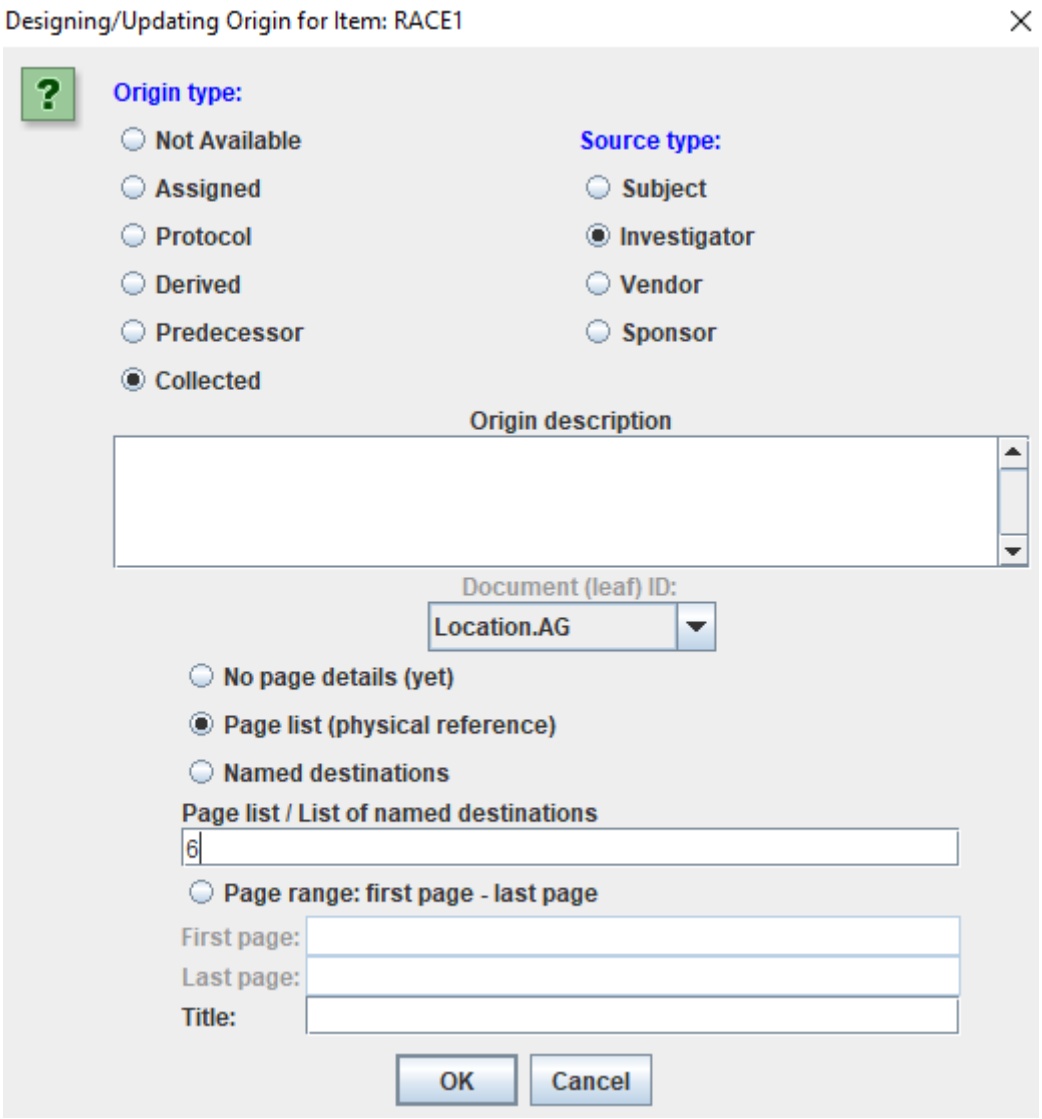
In order to populate QORIG anyway, we just assign an origin to DM.RACE1, DM.RACE2 and DM.RACE3. To do this, we start by selecting the cell DM.RACE1, and then use the menu "Edit - SDTM Variable Properties" (Ctrl-E) and look for the entry "Origin/Source":



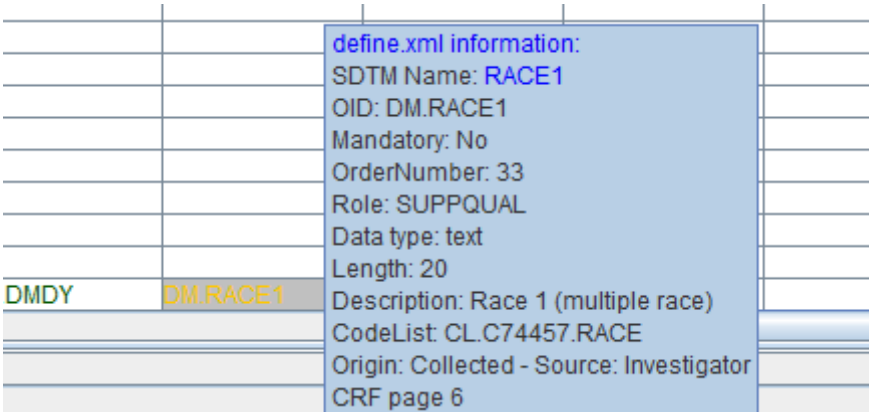
We check the checkbox "Edit Origin/Source" and then click "Edit", leading to:



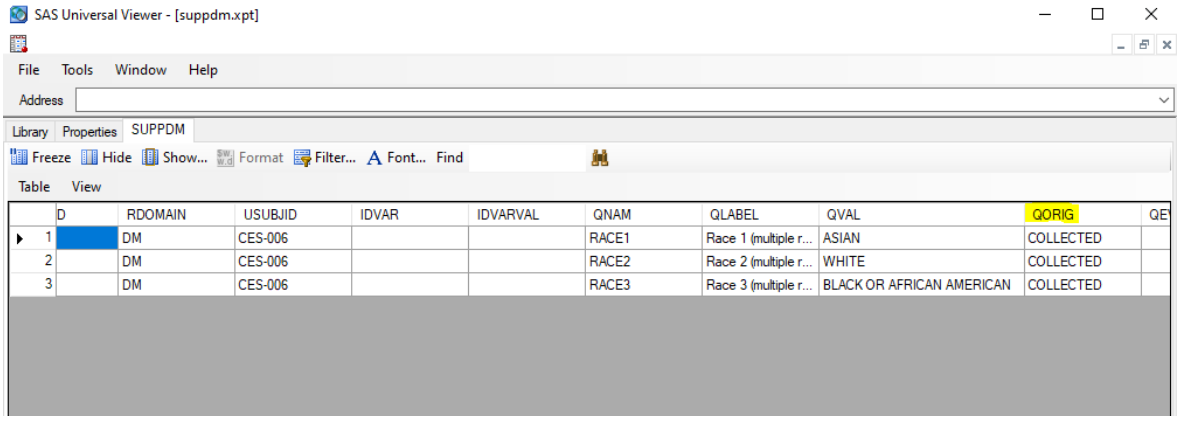
Usually, the values come from the CRF, and was collected by the Investigator, so we select "Collected" and "Investigator", and then add the page number from the annotated CRF, like:



We do the same for the two other NSVs DM.RACE2 and DM.RACE3.
When we then hold the mouse over e.g DM.RACE1, we observe:



When then executing the mappings again, generating SAS-XPT, we get for suppdm.xpt:



	D	RDOMAIN	USUBJID	IDVAR	IDVARVAL	QNAM	QLABEL	QVAL	QORIG	QE
1		DM	CES-006			RACE1	Race 1 (multiple r...	ASIAN	COLLECTED	
2		DM	CES-006			RACE2	Race 2 (multiple r...	WHITE	COLLECTED	
3		DM	CES-006			RACE3	Race 3 (multiple r...	BLACK OR AFRICAN AMERICAN	COLLECTED	

Strings of more than 200 characters

SAS Transport 5 is a very inefficient format. Even though SQL and relational databases already existed in 1999 when the FDA made the use of SAS Transport 5 mandatory, it still made the decision for SAS Transport 5, probably as the only tool the reviewers had was … SAS. Even CSV (Comma Separated Values) would have been a considerably better choice.

One of the limitations of SAS Transport 5 is that string values cannot be longer than 200 characters. If want tries to store a string of more than 200 characters, everything beyond the 200th character will be lost.
Therefore the SDTMIG states that when a string is longer than 200 characters, then the string must be split in chunks of not more than two characters, and the splitting must be done between words. The first chunk then goes into the original variable, and the remaining chunks go into Supplemental Qualifier varibles in the SUPPxx dataset. For example, when the value of MHTERM is e.g. 500 characters. this leads to 3 chunks, the first going into MHTERM, the second going into MHTERM1, and the third in MHTERM2, the second and third then go into SUPPMH. See e.g. p. 55 in the SDTMIG-3.4.

So essentially, SUPPxx datasets are both used for NSVs as well as for "split" variable values when the value is longer than 200 characters.

SDTM-ETL automatically takes care of this situation. So when there are NSVs for a domain and there are values of standard variables longer than 200 characters, SDTM-ETL will take care that both go into the same SUPPxx dataset. An example may be that when there is an NSV in MH providing more specifics on the disease reported, and some values of MHTERM are longer than 200 characters.
Remind that when executing the mappings, one then also needs to check the checkbox "Split records > 200 characters to SUPP-- records"

Remark: the new Dataset-JSON 1.1 format does not have this 200-character limitation, so that no "splitting off" is necessary for the case that Dataset-JSON is used as the output format. The same applies for CSV and SQL output.

Conclusions

SDTM-ETL handles non-standard variables (NSVs) as if they were just normal variables. The software allows to define NSVs and add them to domains. Mappings are defined in the same way as for normal, standard variables. It is only at the very end of the process, that NSVs can be moved to SUPPxx datasets. For testing however, it is recommended to just keep them within the "parent" domain/dataset.
In case of strings longer than 200 characters, and the checkbox "Split records > 200 characters to SUPP-- records" is checked (which is automatically done when selecting SAS-XPT as the output format), then the system automatically takes care that addition records (also when also NSVs are split off) are created in the same SUPPxx dataset.