7 MANIÈRES D'ÉCRIRE UNE PROC SORT

Le tri des données est une manipulation quasi quotidienne pour tout data manager. D'ailleurs, la procédure SORT est la 3ème procédure la plus utilisée sur notre version étudiante en 2014[1]. Cet article détaille la PROC SORT et ses options, puis explique pas-à-pas comment trier ses données dans plusieurs situations (depuis SAS® Enterprise Guide®, depuis SAS® Visual Data Builder, depuis un SGBD etc).

Caractéristiques :

Catégories : Base SAS®
OS : Windows, Unix, z/OS
Version : SAS® 9.4
Vérifié en mai 2015

Sommaire

1. Introduction : dans quels cas faut-il trier les données ? 1
   1.1. Avant une instruction BY 1
   1.2. L'attribut SORTED BY versus l'option PRESORTED 2
   1.3. Pour optimiser un index 2
   1.4. Pour supprimer des données physiquement (versus logiquement) 3
2. Comment trier les observations d'une table 4
   2.1. La procédure SORT 4
      2.1.1. Quid de NODUPKEY, NODUPLICATES, NOEQUALS, NOUNIQUEKEY, NOUNIQUEREC . 4
      2.1.2. OUT, DUPOUT, UNIQUEOUT 7
   2.2. La procédure SQL 9
   2.3. Depuis SAS® Studio 11
      2.3.1. Ecrire son programme 11
      2.3.2. La tâche "Tri des données" 12
   2.4. Depuis SAS Enterprise Guide 17
      2.4.1. Ecrire son programme 17
      2.4.2. La tâche « Tri des données » 20
   2.5. Depuis SAS Data Integration Studio 24
      2.5.1. La transformation « Tri » 25
      2.5.2. La transformation « Extraction » 30
   2.6. Depuis SAS Visual Data Builder 33
   2.7. En code Hash 35
3. In-database processing 36
4. En cas de problème 36
   4.1. Les problèmes les plus fréquents 36
      4.1.1. Manque d'espace disque 36
      4.1.2. Performances 37
   4.2. Eléments à transmettre au Support Clients 37
5. Liens utiles 37

1. INTRODUCTION : DANS QUELS CAS FAUT-IL TRIER LES DONNÉES ?

Il existe bien entendu une multitude de besoins qui vont amener un data manager à effectuer un tri. Mettons en lumière quelques-uns de ces besoins.

1.1. Avant une instruction BY

Que ce soit dans l'ODS Graphics (SGDESIGN, SGPANEL…), dans toutes les procédures graphiques (GPLOT, GMAP…), statistiques (LOGISTIC, SCORE, SHEWART…), de base (PRINT, MEANS, REPORT, TRANSPOSE etc) ou encore dans une étape DATA (MERGE, SET), une instruction BY requiert que la table en entrée soit triée ou indexée, sous peine de recevoir un message du type :

```
ERROR: BY variables are not properly sorted on data set LIBRARY.DSNAME
```

Ou bien

[1] Cf. *Top 20 Procs in SAS University Edition*, 3/12/2014

```
ERROR: Data set LIBRARY.DSNAME is not sorted in ascending sequence
```

L'instruction BY (généralement dénommée « BY-Group Processing » dans notre documentation) permet de spécifier une/des variable(s) de catégories à utiliser pour regrouper les données (par exemple par pays, par sexe…).
Elle accepte toutefois l'option NOTSORTED : si un index existe alors il ne sera pas utilisé, et un tri alphanumérique ne sera pas requis. Chaque nouvelle valeur dans le traitement séquentiel est dès lors traitée comme un nouveau groupe.

Exemple :

Dans cet exemple, l'utilisateur veut une répartition par sexe, et va donc utiliser l'instruction BY de la PROC PRINT. Celle-ci requiert que la table en entrée soit triée.

```
proc sort data=sashelp.class
      out=class ;
  by sex;
  where age ge 15 ;
run ;
proc print data=class noobs
      sumlabel='#byval(sex) Total'
      grandtotal_label='Grand Total';
  by sex;
  var name age height weight;
  sum height weight;
run;
```

Sex=F

Name	Age	Height	Weight
Janet	15	62.5	112.5
Mary	15	66.5	112.0
F Total		129.0	224.5

Sex=M

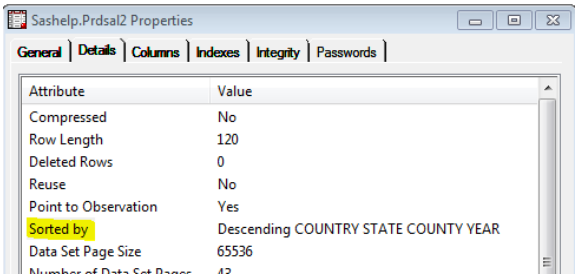
Name	Age	Height	Weight
Philip	16	72.0	150.0
Ronald	15	67.0	133.0
William	15	66.5	112.0
M Total		205.5	395.0
Grand Total		334.5	619.5

A noter : l'instruction WHERE peut être placée dans la PROC SORT ou la PROC PRINT.

1.2. L'attribut SORTED BY versus l'option PRESORTED

Une table est composée d'un en-tête qui la décrit, et de données. Cet en-tête contient un attribut « SORTED BY », qui contient la séquence de tri. Cet attribut est ajouté à la table lors d'un tri, qu'il soit réalisé par la procédure SORT ou SQL.
Si une table est déjà triée par la séquence demandée, alors le tri n'est pas fait. Le journal nous l'indique :

```
NOTE: Input data set is already sorted, no
sorting done.
```



Il peut arriver qu'une table soit triée, mais que l'attribut SORTED BY soit vide : c'est le cas par exemple après avoir importé un fichier plat dont les valeurs étaient triées, ou bien si on a créé une table via une étape DATA en lisant une table triée, et en n'appliquant pas de traitement qui remettrait en cause ce tri.
Dans ce cas, on pourra spécifier l'option PRESORTED au niveau de la PROC SORT. C'est utile lorsque l'on veut profiter d'un traitement sur les doublons par exemple sans que les données soient à nouveau triées. Dans ce cas, SAS s'affranchit du tri et effectue directement le travail spécifique à l'option. On économise du temps et des ressources.

1.3. Pour optimiser un index

L'utilisation d'un index a un certain coût en E/S. Lors de la création d'un index, il peut être judicieux de préalablement trier la table par les variables concernées. Même si ce tri a lui-même un certain coût, c'est à l'utilisation de l'index que le gain se fera ressentir. Il s'agit d'une « bonne pratique » qu'il est important de tester lorsque vous souhaitez créer des indexes. Ce conseil est donné dans

notre documentation, SAS® 9.4 Language Reference: Concepts, chapitre « Understanding SAS Indexes ».

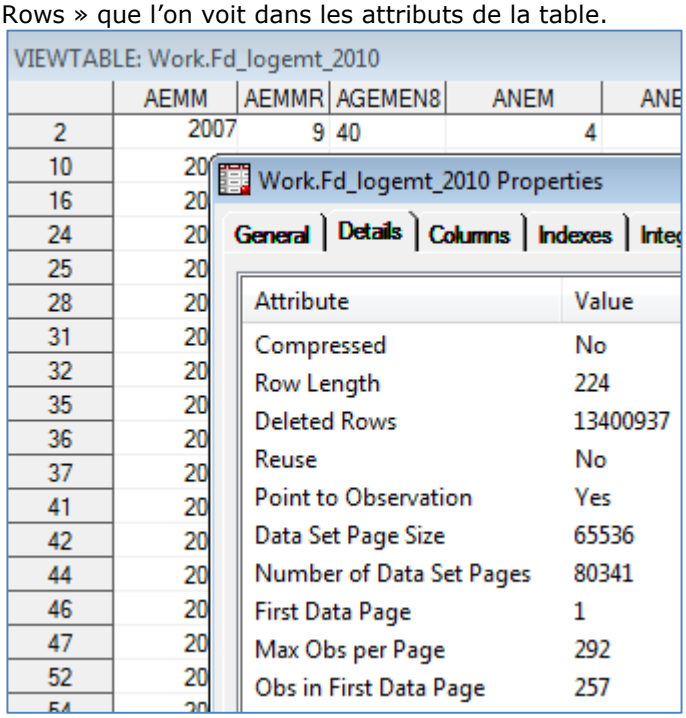
1.4. Pour supprimer des données physiquement (versus logiquement)

Lorsque l'on supprime des observations via l'instruction DELETE de la PROC SQL ou bien en mode FSEDIT, les données sont supprimées logiquement mais elles sont toujours présentes physiquement : elles ne sont plus prises en compte dans les traitements mais elles utilisent toujours de l'espace disque. Ce sont les « Deleted Rows » que l'on voit dans les attributs de la table.

Exemple :
La table WORK.FD_LOGEMNT_2010 contient 23 459 469 observations, et a une taille de 4,9 Go.
Des observations ont été supprimées via la PROC SQL :

```
30   proc sql;
31        delete from FD_LOGEMT_2010
32           where aemm lt 2000 ;
NOTE: 13400937 rows were deleted
from WORK.FD_LOGEMT_2010.
33   quit ;
```

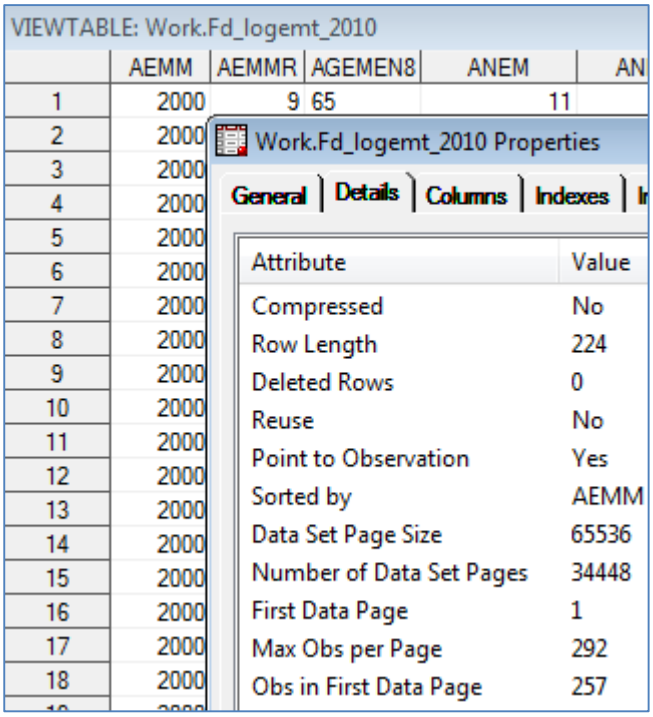
Nous constatons que
- La table occupe le même espace physique : 4,9 Go
- La numérotation des observations n'est pas continue
- L'attribut « deleted rows » indique 13400937 lignes



Plusieurs méthodes permettent de supprimer définitivement ces observations, telles que
- recréer la table depuis une étape DATA ou en PROC SQL
- utiliser la PROC COPY
- trier la table avec la PROC SORT.

```
proc sort data= FD_LOGEMT_2010;
  by AEMM ;
  run ;
```

Après la PROC SORT :
- La table occupe désormais 2,1 Go
- La numérotation des observations est continue
- Le « deleted rows » est désormais de 0, et le nombre d'observations sur la 1ère page est de 5



2. COMMENT TRIER LES OBSERVATIONS D'UNE TABLE

Les exemples donnés dans cette section sont disponibles dans le fichier US2015Q2_PROCSORT.sas.

2.1. La procédure SORT

La procédure SORT permet de trier une table selon une ou plusieurs variables, numériques et caractères, de manière ascendante ou descendante. Elle peut remplacer la table en question, ou bien créer une table en sortie.
Son instruction principale est BY, où l'on va spécifier les variables à trier.
Mais elle accepte également des instructions telles que LABEL, FORMAT ou WHERE qui agiront sur la table en sortie.

Exemple :

```
PROC SORT DATA=Assurance
       OUT=Assurance_F (LABEL="Nb années travaillées
par femme" DROP=SEXE) noduplicates;
  BY NbAnnees  ID ;
  WHERE sexe='F';
  LABEL NbAnnees="Nombre d'années travaillées"
ID="n° d'employé";
RUN ;
```

VIEWTABLE: Work.Assurance_f (Nb ar

	Nombre d'années travaillées	n° d'employé
1	1	564
2	3	343
3	3	711
4	4	616
5	5	421

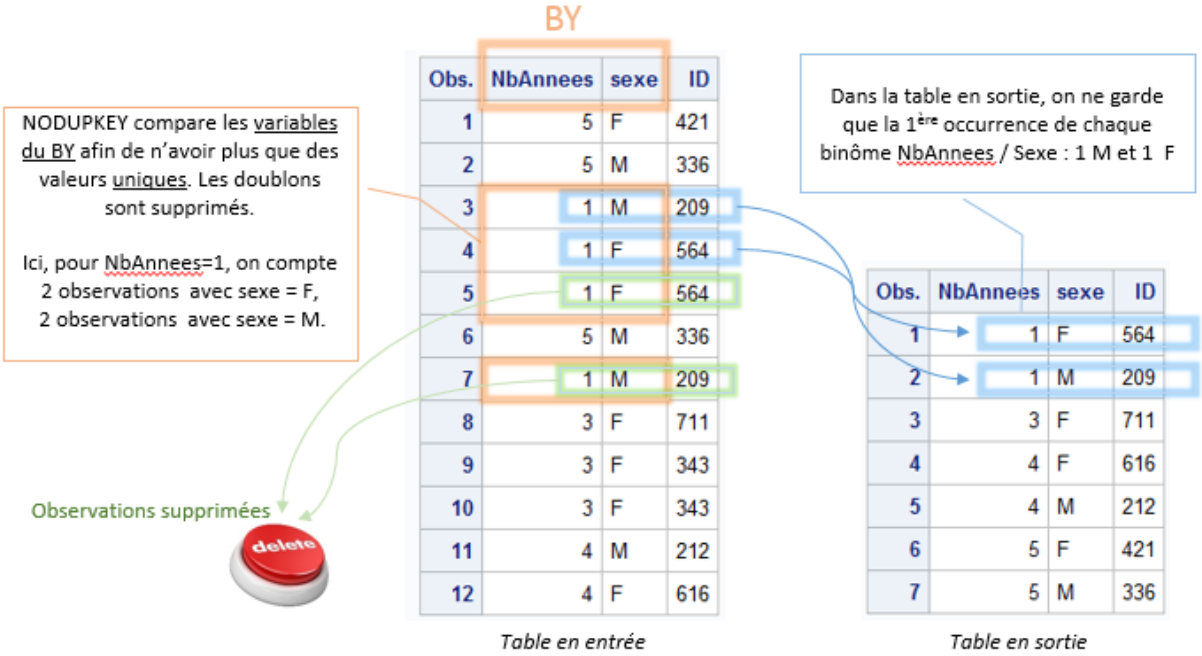
Parmi la multitude d'options possibles, illustrons quelques-unes d'entre elles.

2.1.1. Quid de NODUPKEY, NODUPLICATES, NOEQUALS, NOUNIQUEKEY, NOUNIQUEREC

Ces 5 options influencent le traitement de manière différente. Chaque option sera illustrée avec la table Assurance, créée pour l'occasion, qui contient le nombre d'années travaillées par n° d'employé.

NODUPKEY (alias NODUPKEYS) indique que la clé de tri spécifiée par le BY sera unique dans la table en sortie. Autrement dit, seule la 1ère occurrence du BY sera gardée dans la table en sortie principale, tandis que les doublons sont supprimés (ou bien gardés dans la table spécifiée par DUPOUT, non utilisé dans l'exemple qui suit).
Exemple :

```
proc sort data=Assurance out=ParAnnee NODUPKEY ;
  by NbAnnees sexe ;
run;
```



Obs.	NbAnnees	sexe	ID
1	5	F	421
2	5	M	336
3	1	M	209
4	1	F	564
5	1	F	564
6	5	M	336
7	1	M	209
8	3	F	711
9	3	F	343
10	3	F	343
11	4	M	212
12	4	F	616

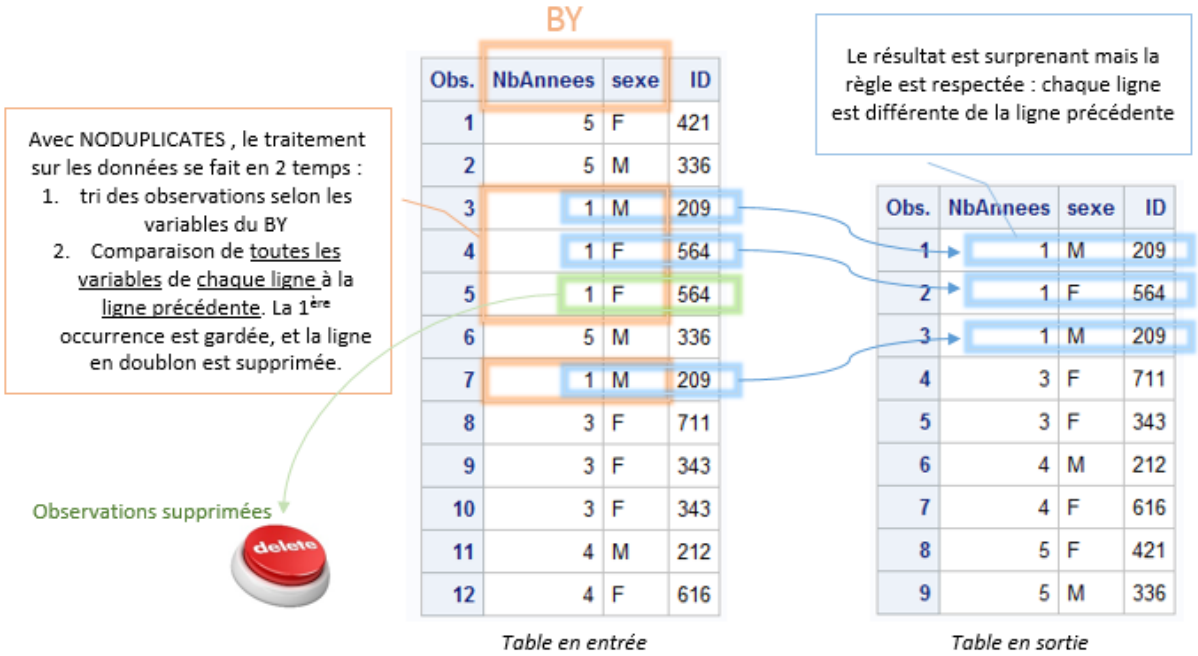
Obs.	NbAnnees	sexe	ID
1	1	F	564
2	1	M	209
3	3	F	711
4	4	F	616
5	4	M	212
6	5	F	421
7	5	M	336

NODUPLICATES[2] (alias NODUP, NODUPLICATE, NODUPREC, NODUPRECS, NODUPS) indique qu'une observation doit être différente de l'observation précédente, une fois le tri effectué. Cela signifie que :

- ce ne sont pas seulement les variables du BY qui sont comparées, mais toutes les variables de la table.
- Une observation n'est pas comparée à l'ensemble des observations de la table, mais seulement à l'observation adjacente.
- Les doublons sont supprimés ou bien gardés dans la table spécifiée par DUPOUT.

Exemple :

```
proc sort data=Assurance out=ParAnnee NODUPLICATES ;
  by NbAnnees;
run;
```



Obs.	NbAnnees	sexe	ID
1	5	F	421
2	5	M	336
3	1	M	209
4	1	F	564
5	1	F	564
6	5	M	336
7	1	M	209
8	3	F	711
9	3	F	343
10	3	F	343
11	4	M	212
12	4	F	616

Obs.	NbAnnees	sexe	ID
1	1	M	209
2	1	F	564
3	1	M	209
4	3	F	711
5	3	F	343
6	4	M	212
7	4	F	616
8	5	F	421
9	5	M	336

A noter : cette option est dépréciée et pourrait disparaître dans une version future.

EQUALS indique que les valeurs des variables qui ne font pas partie du BY doivent garder l'ordre qu'elles ont à l'origine dans la table. NOEQUALS indique que les valeurs des variables peuvent être dans un ordre différent, c'est d'ailleurs l'option qui est appliquée par défaut lorsqu'EQUALS n'est pas précisée. Autrement dit, par défaut l'ordre des observations au sein d'un même groupe BY de tri n'est pas garanti. Si on souhaite retrouver l'ordre d'origine, il faut spécifier l'option EQUALS au niveau de PROC SORT, ou bien spécifier l'option SORTEQUALS au niveau de la session SAS (instruction « Options », fichier de configuration).

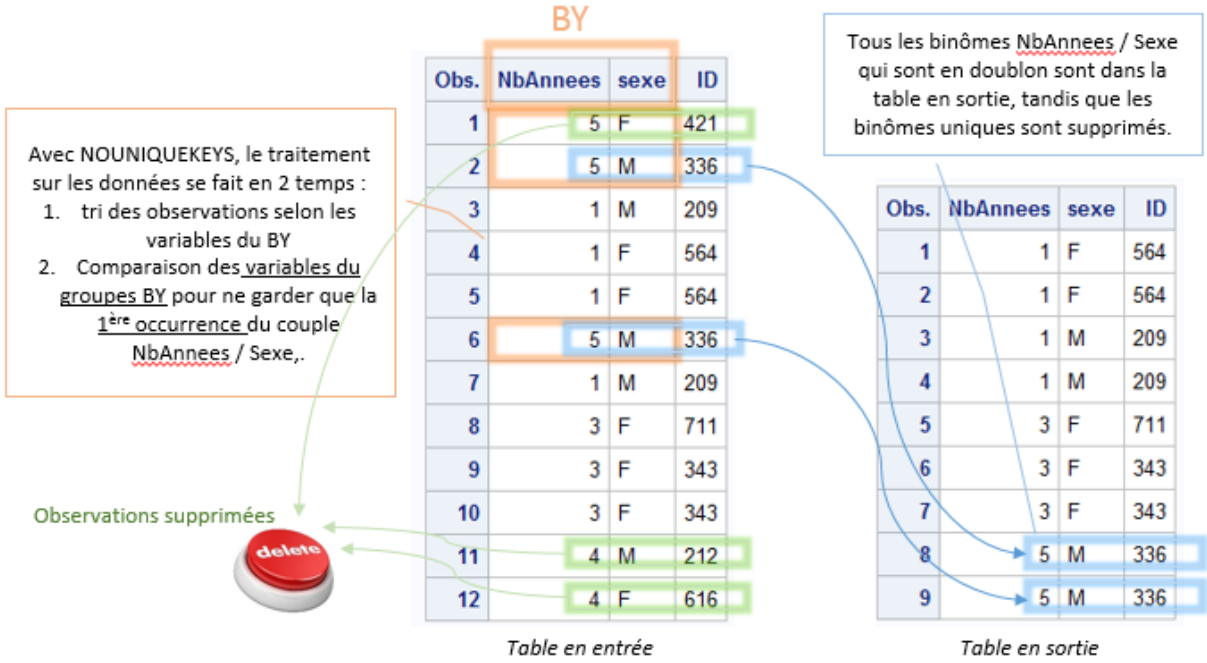
Exemple :

```
proc sort data=Assurance out=ParAnnee_equals EQUALS ;
  by NbAnnees Sexe ;
run;
```

[2] Cette option est dépréciée depuis SAS 9.3 M2. Elle est gardée pour des raisons de compatibilité mais pourrait disparaître dans une version future.

NOUNIQUEKEY (alias NOUNIKEY, NOUNIKEYS, NOUNIQUEKEYS) indique que la clé de tri spécifiée par le BY ne doit pas être unique dans la table en entrée. Si la clé est unique, alors l'observation est supprimée de la table en sortie ou bien gardée dans la table spécifiée par UNIQUEOUT.
Exemple :

```
proc sort data=Assurance out=ParAnnee NOUNIQUEKEY  ;
  by NbAnnees Sexe ;
run;
```

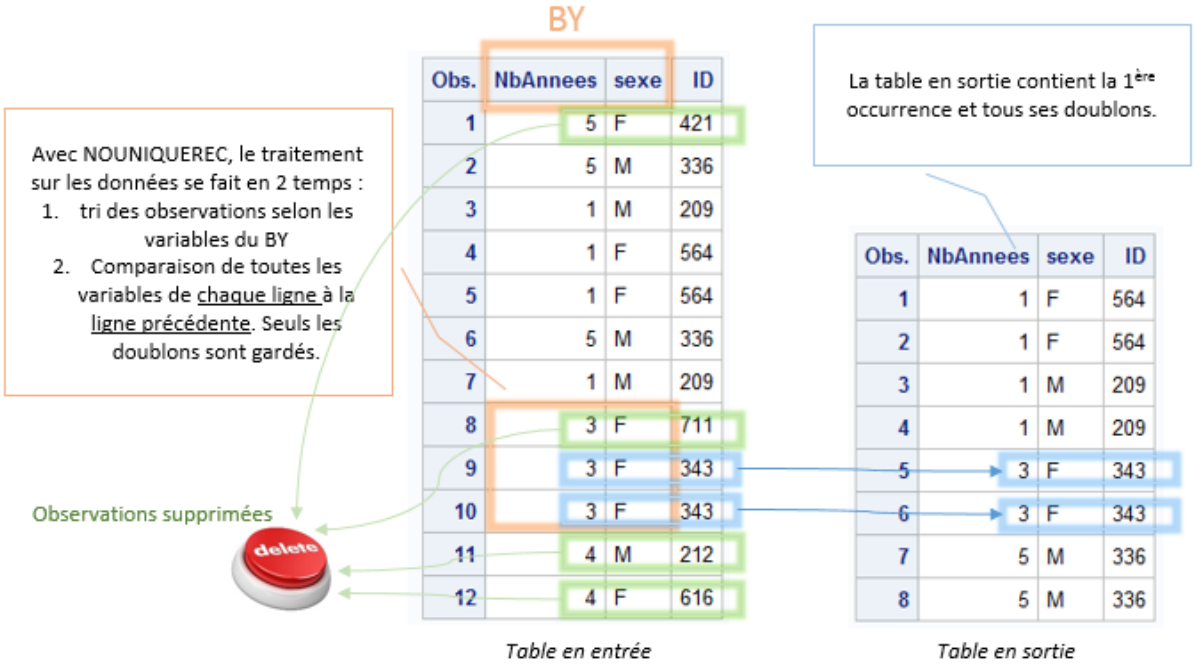


NOUNIQUEREC[3] (alias NOUNIQUERECS, NOUNIREC, NOUNIRECS) indique qu'une observation ne doit pas être différente de la suivante, après que le tri ait été réalisé. C'est l'option inverse de NODUPLICATES : ici on ne veut garder que les lignes en doublons. Si l'observation est différente de la précédente, alors elle est supprimée de la table en sortie ou bien gardée dans la table spécifiée par UNIQUEOUT.

[3] Cette option est dépréciée depuis SAS 9.3 M2. Elle est gardée pour des raisons de compatibilité mais pourrait disparaitre dans une version future.

Exemple :

```
proc sort data=Assurance out=ParAnnee NOUNIQUEREC  ;
  by NbAnnees Sexe ;
run;
```



Obs.	NbAnnees	sexe	ID
1	5	F	421
2	5	M	336
3	1	M	209
4	1	F	564
5	1	F	564
6	5	M	336
7	1	M	209
8	3	F	711
9	3	F	343
10	3	F	343
11	4	M	212
12	4	F	616

Obs.	NbAnnees	sexe	ID
1	1	F	564
2	1	F	564
3	1	M	209
4	1	M	209
5	3	F	343
6	3	F	343
7	5	M	336
8	5	M	336

2.1.2. OUT, DUPOUT, UNIQUEOUT

Ces trois options concernent les tables créées en sortie de la PROC SORT. Elles acceptent les options habituelles que l'on peut utiliser lors de la création d'une table telle que RENAME, DROP, KEEP, WHERE, LABEL.
Ce paragraphe sera illustré avec la table SASHELP.CARS disponible sur chaque installation SAS.

OUT désigne la table en sortie contenant les données triées : la table en entrée ne subit aucun changement, seule la table en sortie sera triée. Si OUT n'est pas spécifiée, alors la table d'origine est écrasée avec les observations triées.

Exemple :

```
proc sort   data=sashelp.cars
            out=cars (drop=MPG: Origin where=(cylinders=6));
  by make Type ;
run ;
```

La log nous indique que la table WORK.CARS a été créée :

```
NOTE: There were 428 observations read from the data set SASHELP.CARS.
NOTE: The data set WORK.CARS has 190 observations and 12 variables.
```

DUPOUT permet de créer une seconde table en sortie qui contiendra les observations qui n'auront pas été gardées suite à l'utilisation de NODUPKEY ou NODUPLICATES. Elle s'utilise donc en combinaison

- Soit avec NODUPKEY, qui indique que la clé de tri spécifiée par le BY sera unique dans la table en sortie. Autrement dit, la 1ère occurrence de valeurs est gardée dans la table en sortie principale, tandis que les doublons sont mis dans la table DUPOUT.
- Soit avec NODUPLICATES, qui indique que l'ensemble de l'observation doit être différente de l'observation présente, une fois le tri effectué. Ce ne sont pas seulement les variables du BY qui sont comparées, mais toutes les variables de la table.

Exemple :
La table SASHELP.CARS contient 7 observations pour MAKE='Acura', avec plusieurs valeurs pour TYPE :

VIEWTABLE: Sashelp.Cars (2004 Car Data)

	Make	Model	Type	Origin	Driv
1	Acura	MDX	SUV	Asia	All
2	Acura	RSX Type S 2dr	Sedan	Asia	Fro
3	Acura	TSX 4dr	Sedan	Asia	Fro
4	Acura	TL 4dr	Sedan	Asia	Fro
5	Acura	3.5 RL 4dr	Sedan	Asia	Fro
6	Acura	3.5 RL w/Navigation 4dr	Sedan	Asia	Fro
7	Acura	NSX coupe 2dr manual S	Sports	Asia	Rea
8	Audi	A4 1.8T 4dr	Sedan	Europe	Fro

On la trie avec NODUPKEY et DUPOUT :

```
proc sort data=sashelp.cars out=cars DUPOUT=cars_dupout
nodupkey;
  by make Type ;
run ;
```

La log nous indique ceci :

```
NOTE: The data set WORK.CARS has 114 observations and 15 variables.
NOTE: The data set WORK.CARS_DUPOUT has 314 observations and 15 variables.
```

En sortie, la table WORK.CARS ne contient plus que 3 observations pour MAKE='Acura'. Le binôme MAKE / TYPE est unique :

	Make	Model	Type	C
1	Acura	MDX	SUV	Asi
2	Acura	RSX Type S 2dr	Sedan	Asi
3	Acura	NSX coupe 2dr manual S	Sports	Asi
4	Audi	A4 1.8T 4dr	Sedan	Eu
5	Audi	RS 6 4dr	Sports	Eu

La table WORK.CARS_DUPOUT contient les 4 observations qui n'ont pas été gardées dans WORK.CARS car le binôme MAKE / TYPE était en doublon:

	Make	Model	Type
1	Acura	TSX 4dr	Sedan
2	Acura	TL 4dr	Sedan
3	Acura	3.5 RL 4dr	Sedan
4	Acura	3.5 RL w/Navigation 4dr	Sedan
5	Audi	A41.8T convertible 2dr	Sedan

<u>UNIQUEOUT</u> permet de créer une seconde table en sortie qui contiendra les observations qui ont été éliminées :

- Soit par l'option NOUNIQUEKEY, qui indique que la clé de tri spécifiée par le BY ne doit pas être unique dans la table en sortie
- Soit par l'option NOUNIQUEREC qui indique que l'ensemble de l'observation ne doit pas être unique.

Exemple :
On trie SASHELP.CARS avec NODUPKEY et DUPOUT :

```
proc sort data=sashelp.cars out=cars UNIQUEOUT=cars_UNIQUEOUT
noUNIQUEkey;
  by make Type ;
run ;
```

La log nous indique :

```
NOTE: The data set WORK.CARS has 389 observations and 15 variables.
```

```
NOTE: The data set WORK.CARS_UNIQUEOUT has 39 observations and 15 variables.
```

En sortie, la table WORK.CARS contient 5 lignes pour MAKE='ACURA', où l'on note qu'il y a plusieurs lignes pour le même binôme MAKE / TYPE :

	Make	Model	Type
1	Acura	RSX Type S 2dr	Sedan
2	Acura	TSX 4dr	Sedan
3	Acura	TL 4dr	Sedan
4	Acura	3.5 RL 4dr	Sedan
5	Acura	3.5 RL w/Navigation 4dr	Sedan
6	Audi	A4 1.8T 4dr	Sedan

En revanche, la table CARS_UNIQUEOUT contient les observations où le binôme MAKE/TYPE est unique :

	Make	Model	Type
1	Acura	MDX	SUV
2	Acura	NSX coupe 2dr manual S	Sports
3	BMW	325xi Sport	Wagon

2.2. La procédure SQL

La procédure SQL peut également être utilisée pour trier une table, via l'instruction ORDER BY.

Exemple :

```
PROC SQL ;
  CREATE TABLE Assurance_F (LABEL="Nb années
travaillées par femme") as
  SELECT DISTINCT NbAnnees LABEL="Nombre
d'années travaillées", ID LABEL="n°
d'employé"
  FROM Assurance
  WHERE sexe='F'
  ORDER BY NbAnnees, ID ;
  QUIT ;
```

VIEWTABLE: Work.Assurance_f (Nb ar

	Nombre d'années travaillées	n° d'employé
1	1	564
2	3	343
3	3	711
4	4	616
5	5	421

L'avantage d'utiliser la procédure SQL tient principalement au fait d'économiser des étapes : par exemple pour obtenir une statistique sur un groupe trié, on utilisera plusieurs étapes via les procédures SAS/Base habituelles, tandis que cela est peut-être réalisable en une seule PROC SQL. Si l'utilisateur a des traitements spécifiques à exécuter, l'étape DATA propose toutefois un panel de traitements plus importants.
La procédure SORT propose pour sa part des options propres aux doublons, qu'on ne retrouve pas en SQL.

Notons également que l'ordre des valeurs dans la table en sortie n'est garanti que pour les variables de tri : l'ordre initial ne sera pas forcément retrouvé pour les autres variables. C'est également vrai pour la procédure SORT, sauf lorsque l'option EQUALS est utilisée. Ce comportement est décrit dans ce document en ligne.

On peut souvent lire que l'option DISTINCT de la PROC SORT est l'équivalent de l'option NODUPLICATES de la PROC SORT, mais c'est une erreur :

- NODUPLICATES compare une observation à l'observation précédente (lecture séquentielle de la table après le tri)
- DISTINCT compare une observation à la totalité des observations.

Exemple :

La table d'origine se présente comme ceci :

VIEWTABLE: Work.Assurance (Nombre d'années trav

	NbAnnees	sexe	ID
1	5	F	421
2	5	M	336
3	1	M	209
4	1	F	564
5	1	F	564
6	5	M	336
7	1	M	209

Nous allons la trier avec la procédure SORT et la procédure SQL. L'une retournera 5 observations, l'autre 4 observations.

Celle-ci retourne 5 observations :

```
proc sort data=Assurance
out=ParAnnee_sort NODUPLICATES ;
  by NbAnnees  ;
run;
```

VIEWTABLE: Work.Parannee_sort (Nombre d'anné

	NbAnnees	sexe	ID
1	1	M	209
2	1	F	564
3	1	M	209
4	5	F	421
5	5	M	336

(pour une illustration de ce phénomène, voir le paragraphe NODUPLICATES).

Tandis que celle-ci retourne 4 observations :

```
PROC SQL ;
  CREATE TABLE ParAnnee_SQL
  SELECT distinct *
  FROM Assurance
  ORDER BY NbAnnees ;
  QUIT ;
```

VIEWTABLE: Work.Parannee_sql

	NbAnnees	sexe	ID
1	1	F	564
2	1	M	209
3	5	F	421
4	5	M	336

2.3. Depuis SAS® Studio

Depuis SAS Studio, l'utilisateur peut trier ses données de 2 façons :

- Il peut écrire manuellement son programme et l'exécuter ;
- Il peut également choisir la tâche « Tri des données ». C'est un assistant pour écrire une PROC SORT : au fur et à mesure des choix qui sont faits, la procédure s'écrit dans la fenêtre « Code ».

Nous allons illustrer ces 2 méthodes, en reproduisant cette requête :

```
PROC SORT DATA=sashelp.class
      OUT=class_13
      (LABEL="Taille des filles -13 ans" KEEP=name height);
  BY name height ;
  WHERE age le 13 and sex='F';
  LABEL name="Prénom" height="Taille";
RUN ;
```

2.3.1. Ecrire son programme

Choisir Nouveau > Programme SAS pour qu'un nouveau programme vierge apparaisse

Puis écrire son programme :

Dans ce mode, si l'utilisateur veut consulter sa table en sortie de manière interactive (voir ci-dessous), il ajoutera ceci à la fin de son programme :

```
%web_open_table(NomDeLaBibliothèque.NomDeLaTable);
```

Quelques actions possibles depuis la barre d'outils :

Créer un nouveau programme SAS

Ajouter à mes extraits de code

Enregistrer et Enregistrer sous

Passer en mode interactif

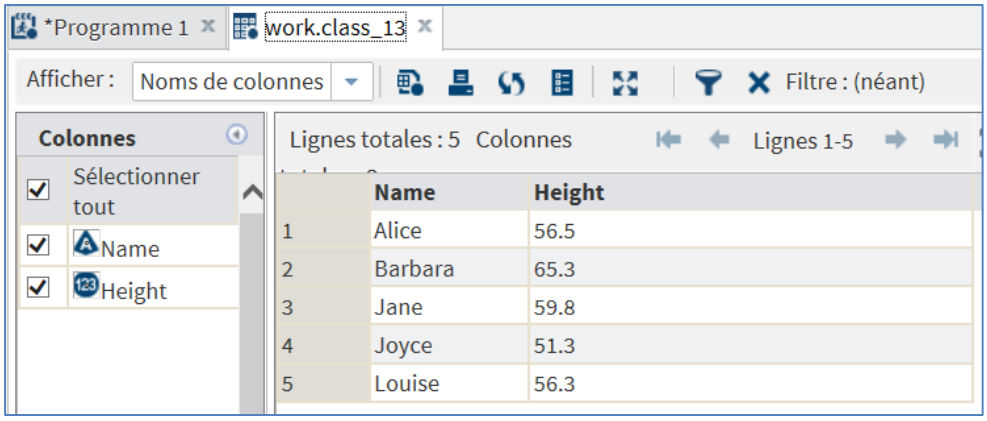
Formatage du code : applique automatiquement les indentations facilitant la lecture du programme

Couper, copier, coller

Supprimer tout le code

Aller à une ligne, Rechercher et remplacer

Le code peut ensuite être exécuté, soit via la touche F3, soit en cliquant sur l'icône habituelle . L'utilisateur pourra accéder au journal. Il pourra également visualiser la table, via une interface interactive :

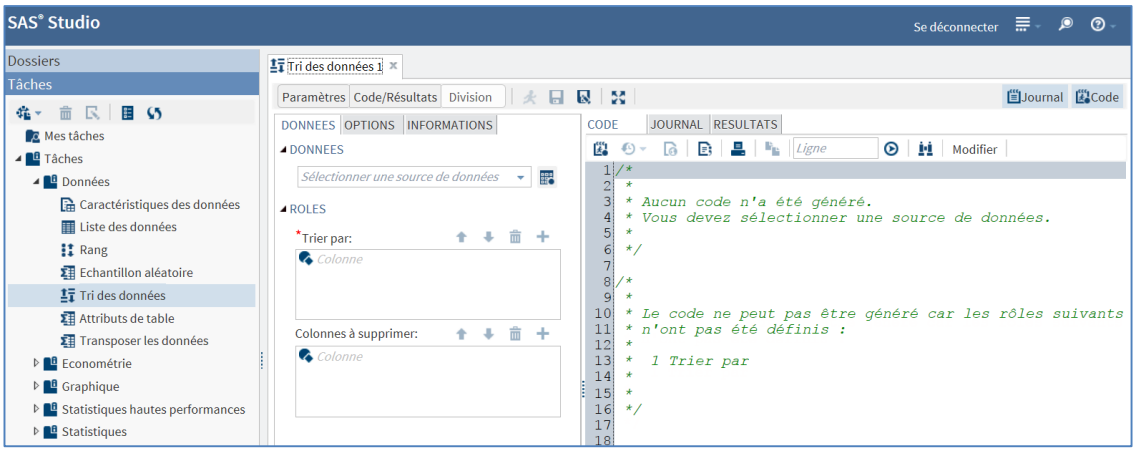


2.3.2. La tâche "Tri des données"

La tâche « Tri des données » est disponible sous Tâches > Données. C'est un assistant pour trier une table via la PROC SORT : au fur et à mesure des choix qui sont faits, la procédure s'écrit dans la fenêtre « Code ». Il faut noter que, bien que le code soit visible, il ne peut pas être édité par l'utilisateur : il ne sera modifié que via l'assistant.

Il faut d'abord sélectionner la tâche :

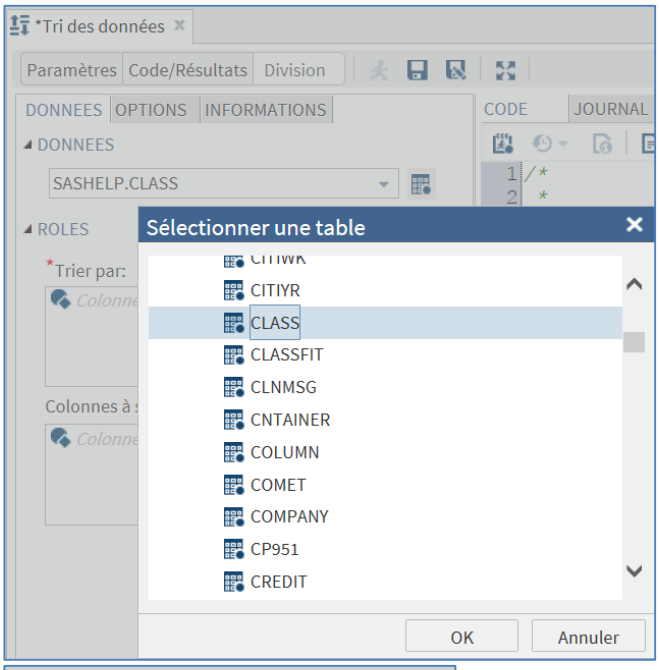
Puis la glisser dans la zone de travail ou blien double-cliquer dessus :



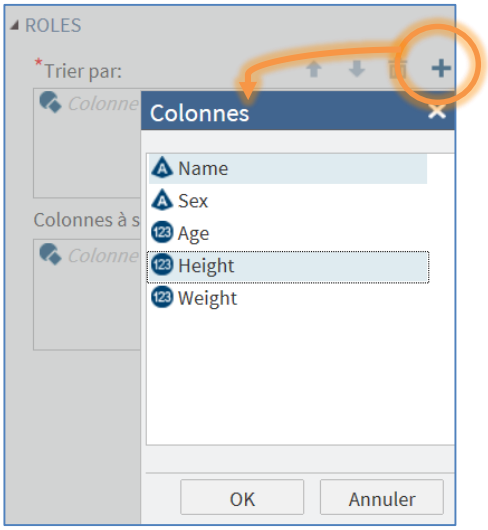
Cette tâche se découpe en 3 onglets :

- Données : pour spécifier la table en entrée (DATA=), les variables à trier (BY), et les variables à ne pas garder dans la table en sortie (DROP)
- Options : pour ajouter des options à l'instruction PROC SORT telles que NODUP, TAGSORT etc.
- Informations : pour accéder à une description de la tâche et l'aide en ligne.

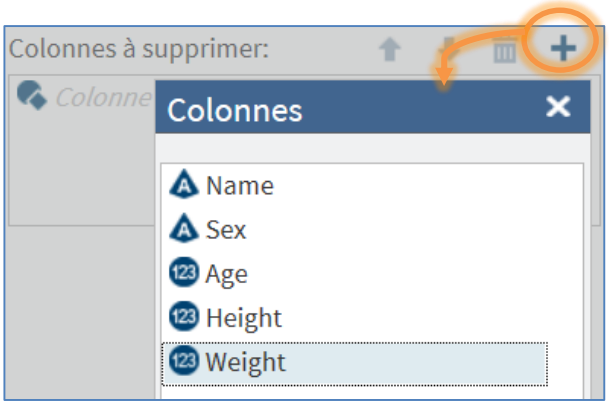
Ensuite l'utilisateur sélectionne la table qu'il veut trier :



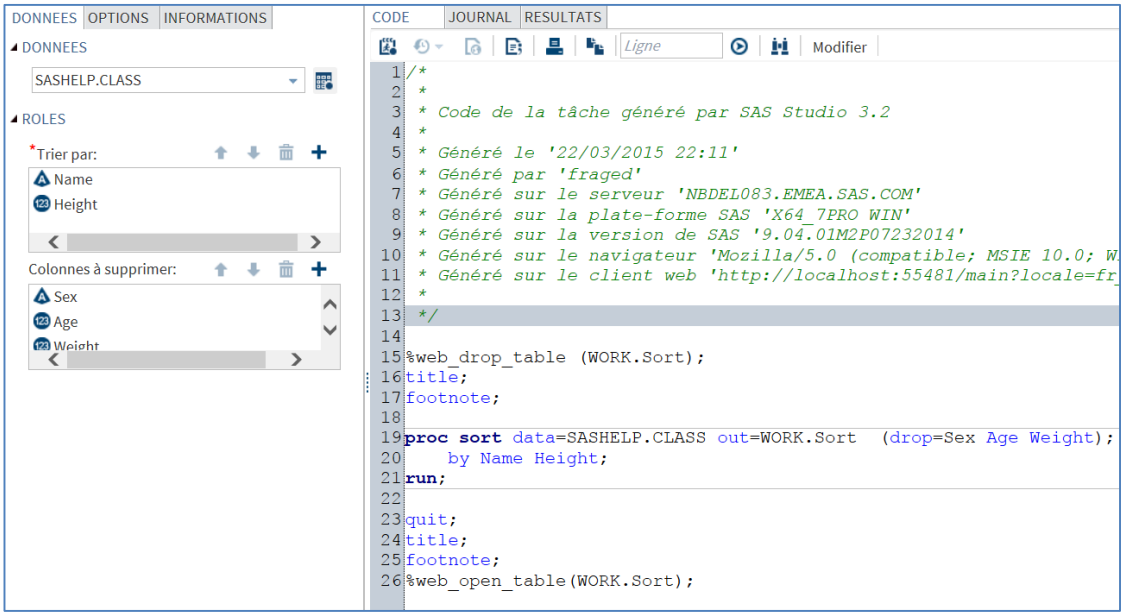
Puis il peut sélectionner les variables, soit une à une en cliquant de nouveau sur le signe +, soit en gardant la touche Shift enfoncée pendant qu'il clique sur plusieurs variables :



Facultatif : il peut aussi choisir des variables à ne pas garder dans la table en sortie, sachant qu'il pourra également masquer des colonnes lors de la consultation du résultat.



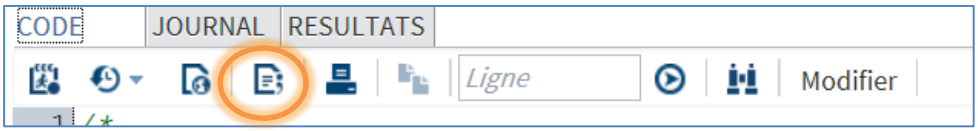
Le code s'écrit au fur et à mesure que les champs sont sélectionnés, il est visible depuis l'onglet « Code » :



Ce code est généré automatiquement et ne peut pas être modifié manuellement.

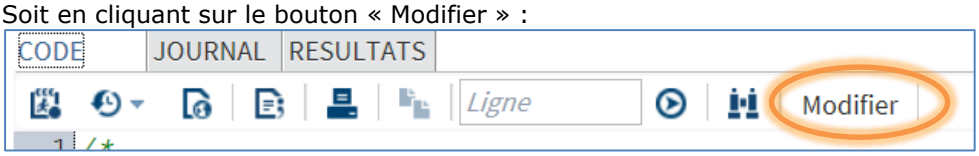
Si l'utilisateur souhaite l'enregistrer pour le modifier manuellement, il a 2 possibilités :

- Soit en créant un extrait de code :



  - Il doit cliquer sur le bouton « Ajouter à mes extraits de code »
  - Lui donner un nom
  - Ensuite le programme s'affichera dans un nouvel onglet, et sera devenu indépendant de la tâche. Il pourra être modifié manuellement.

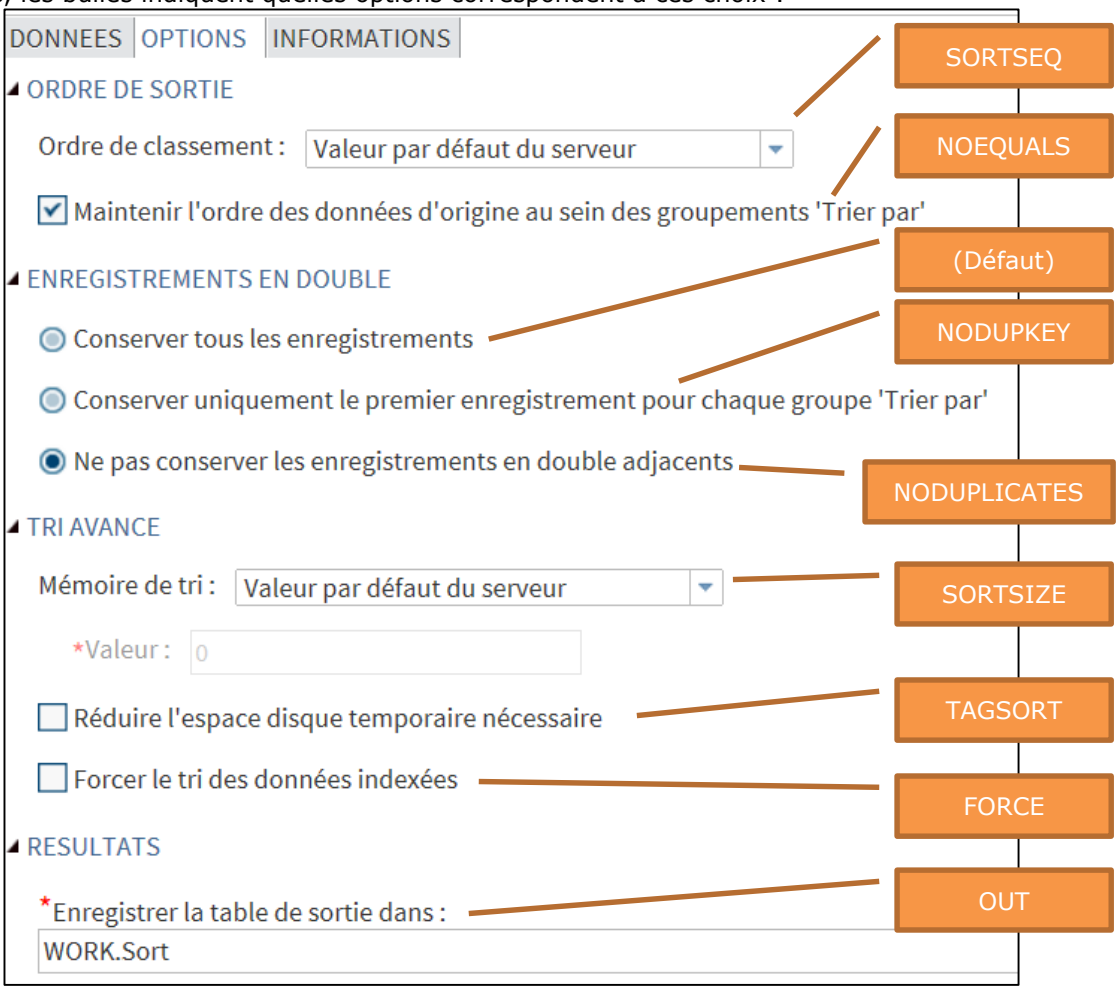
- Soit en cliquant sur le bouton « Modifier » :



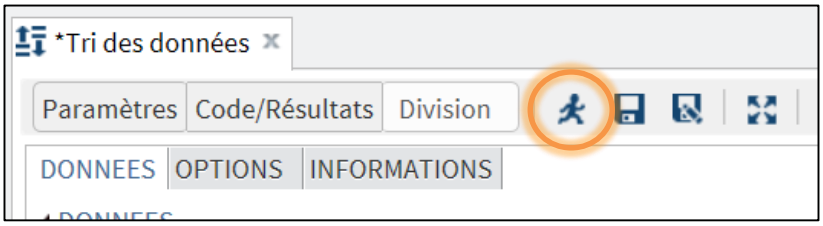
Le programme est éditable, et devra être sauvegardé physiquement (.sas).

Ces 2 actions n'auront pas d'effet sur la tâche de tri elle-même : ce code est indépendant sur la tâche. Par exemple, si l'utilisateur ajoute une instruction LABEL dans l'extrait de code créé, alors celle-ci n'apparaîtra pas dans le code associé à la tâche.

L'utilisateur peut soumettre son code, ou bien aller dans l'onglet « Options ». Lorsque des choix sont faits, des options sont ajoutées dans le code au niveau de l'instruction PROC SORT. Dans l'image ci-dessous, les bulles indiquent quelles options correspondent à ces choix :



Le code peut ensuite être exécuté, soit via la touche F3, soit en cliquant sur l'icône habituelle :



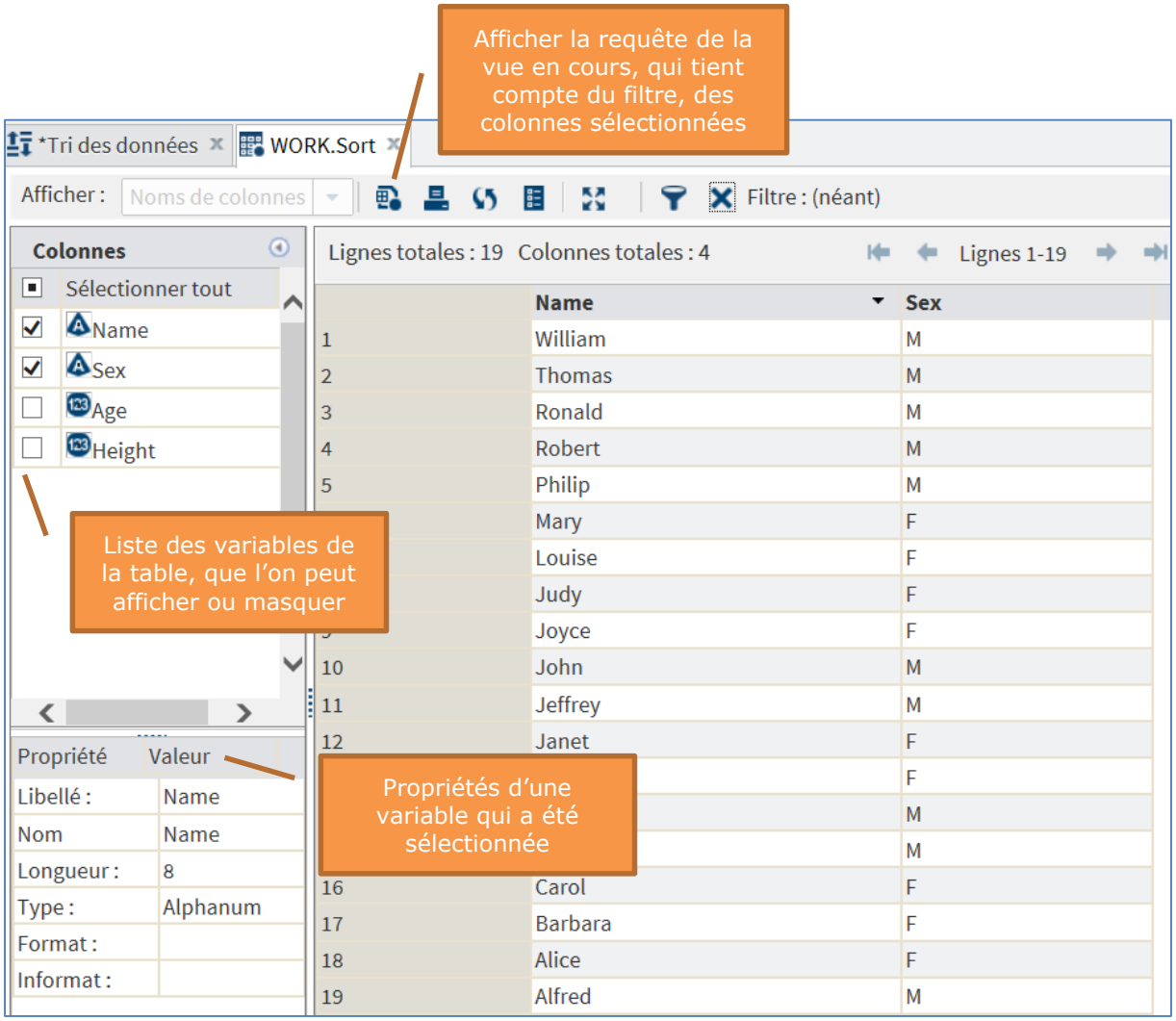
Ce sont 2 actions distinctes qui sont lancées :

- Exécution de la procédure de tri pour créer la table physiquement,
- Puis exécution d'une PROC SQL similaire qui permettra l'affichage en mode interactif de la table.

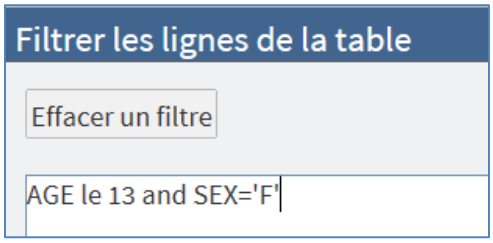
Le journal est accessible. En même temps, la table résultat s'affiche via un nouvel onglet, en mode interactif.

Cette fenêtre est interactive, et offre plusieurs actions à l'utilisateur. Il lui suffira de cliquer, sans avoir besoin de relancer le code initial :

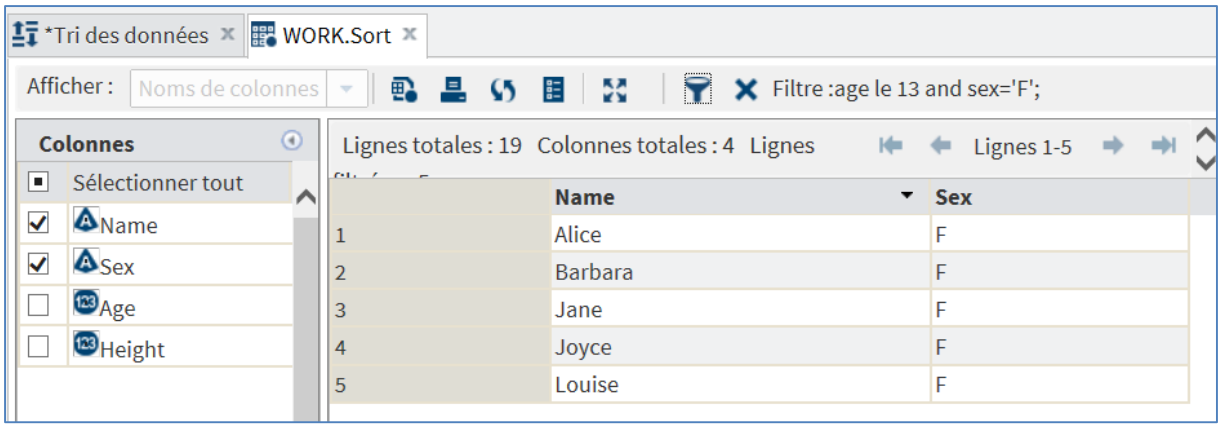
- choisir les colonnes à afficher ou à masquer
- appliquer un fitre
- imprimer la table
- afficher les propriétés des colonnes et de la table
- afficher le code qui est exécuté pour permettre cet affichage.



Pour notre exemple, il nous reste à ajouter un filtre. Il faut cliquer sur l'icône , puis saisir la requête adéquate. Ce filtre est appliqué sur la consultation de la table mais n'a pas d'impact sur la table physique : si l'on ouvre cette table ou que l'on fait un traitement dessus, il n'y aura pas de filtre.



Le filtre est appliqué directement sur la table, sans avoir à relancer le code :



On notera qu'en restant totalement dans la méthode « clic souris », l'utilisateur ne peut pas :

- ajouter un label à la table en sortie
- modifier les attributs d'une variable (libellé, longueur, format, informat)
- utiliser des options telles que DUPOUT ou NOUNIQUEKEY
- appliquer un filtre sur la table en sortie.

2.4. Depuis SAS Enterprise Guide

Comme pour SAS Studio, l'utilisateur peut trier ses données de 2 façons :

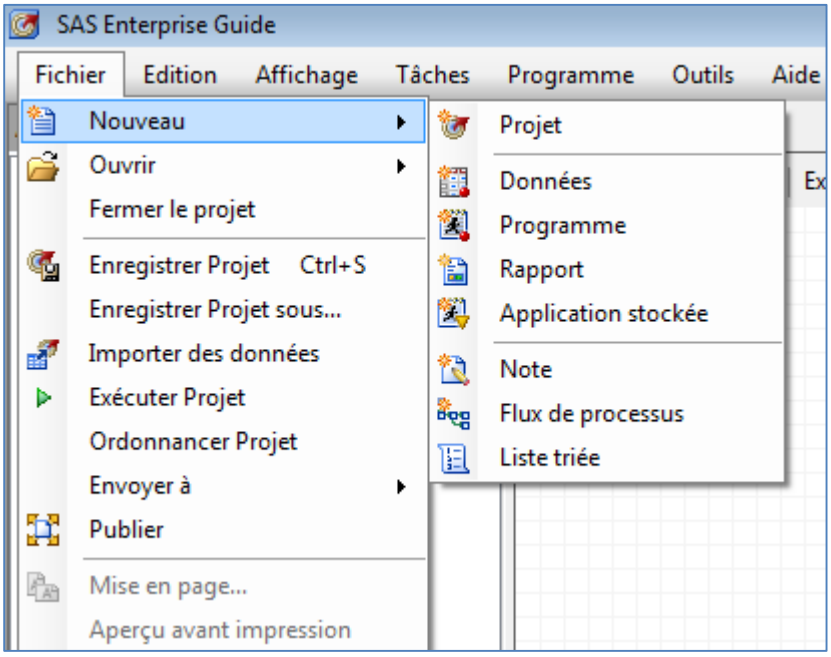
- Il peut écrire manuellement son programme et l'exécuter ;
- Il peut également choisir la tâche « Tri des données ». C'est un assistant pour générer la procédure SORT.

Nous allons illustrer ces 2 méthodes, en reproduisant toujours la même requête :

```
PROC SORT DATA=sashelp.class
      OUT=class_13
      (LABEL="Taille des filles -13 ans" KEEP=name height);
  BY name height ;
  WHERE age le 13 and sex='F';
  LABEL name="Prénom" height="Taille";
RUN ;
```

2.4.1. Ecrire son programme

Pour ouvrir un nouveau programme, aller dans le menu Fichier > Nouveau > Programme



Puis écrire son programme :

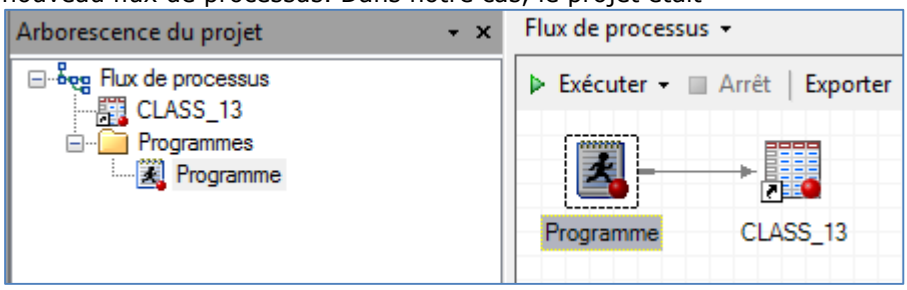
Programme*

Enregistrer ▾ Exécuter ▾ Arrêt | Serveur sélectionné : SASApp (Connecté) ▾ | Analyser programme ▾ | Export

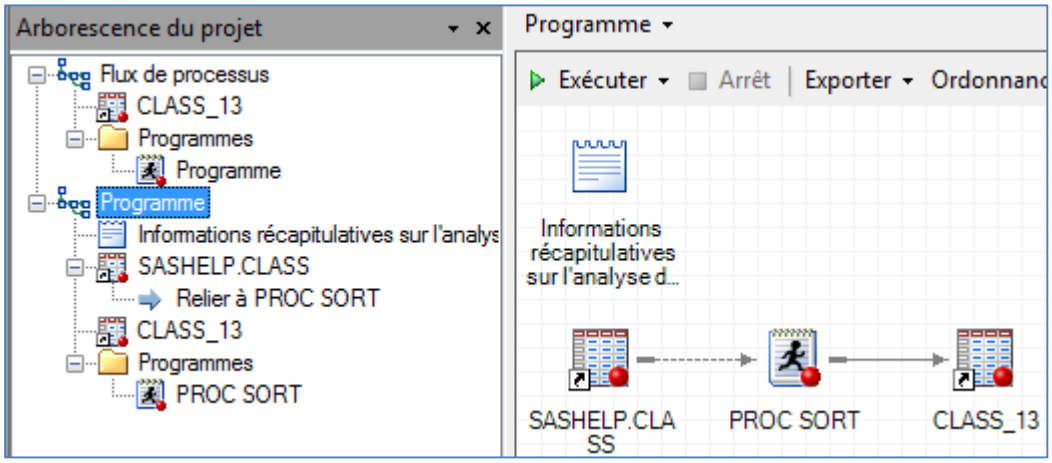
```
PROC SORT DATA=sashelp.class
      OUT=class_13 (LABEL="Taille des filles -13 ans" KEEP=name height);
   BY name height ;
   WHERE age le 13 and sex='F';
   LABEL name="Prénom" height="Taille";
 RUN ;
```

Cette fenêtre permet diverses actions, telles que :

- Analyser le programme pour ajout au flux de programmes, pour le grid ou pour l'internationalisation : l'interface vérifie que les étapes clés du code sont adaptées au grid ou à l'internationalisation.
    - Analyse pour flux de programmes : la procédure SCAPROC analyse le code de l'utilisateur et le transforme en tâches individuelles, que l'on peut ajouter dans un nouveau flux de processus. Dans notre cas, le projet était

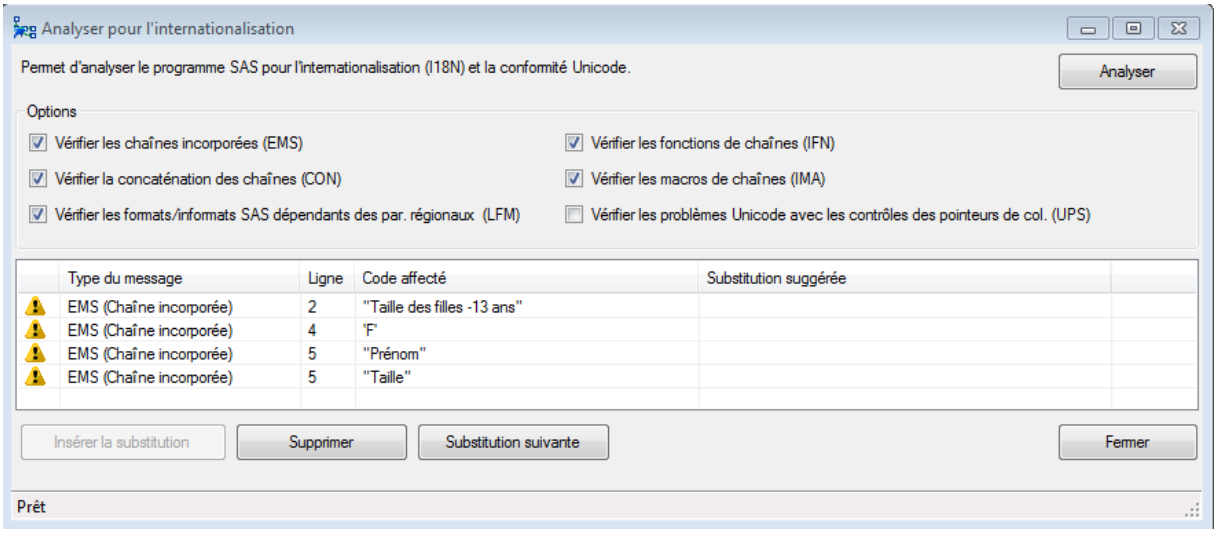


Il devient :



    - Analyse pour Grid Computing : les instructions nécessaires à l'exécution via le grid sont ajoutées au programme (grdsvc_enable…), que l'on peut ensuite ajouter au projet indépendamment du programme d'origine.

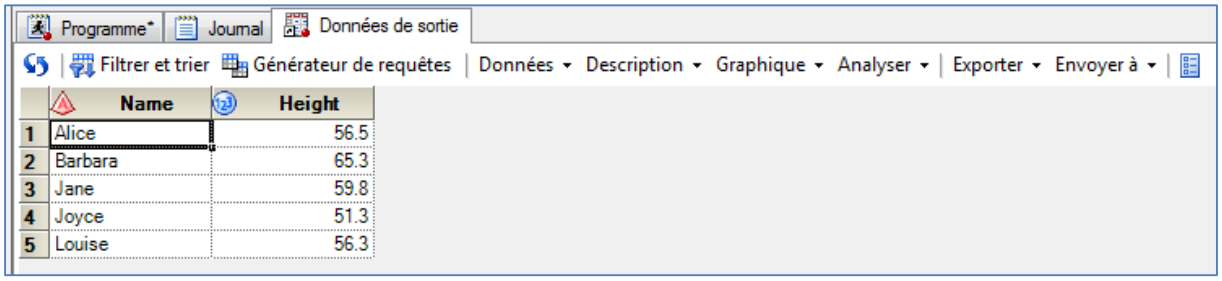
    - Analyse pour l'internationalisation : le but est de vérifier que le code est conforme à la norme I18N et pourra être soumis en environnement Unicode. Dans notre cas, il est suggéré de remplacer les accents et les guillemets :



- Exporter : pour sauver le code dans un fichier de type .sas, en ajoutant les instructions supplémentaires qu'Enterprise Guide soumet
- Créer une application stockée

L'exécuter via les clés F3, F8 ou en cliquant sur le bouton « Exécuter » Exécuter .
En résultat :

- Le journal est accessible.
- La table en sortie est ajoutée au « flux de processus ».
- La table en sortie s'affiche via l'onglet « Données de sortie », et propose des actions à réaliser sur la table :



Depuis cet onglet, l'utilisateur peut réaliser une multitude d'actions, telle que :

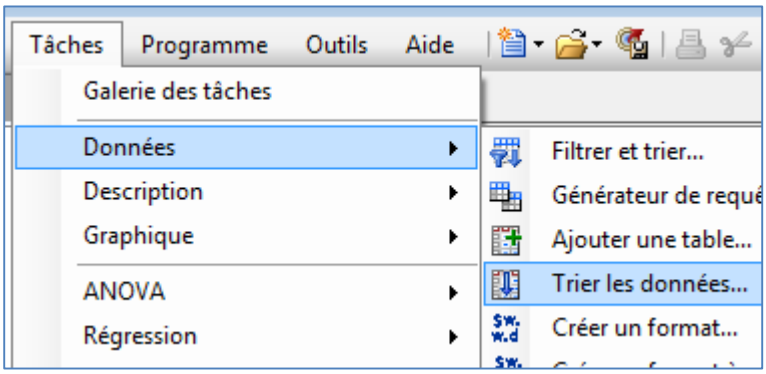
- Filtrer et trier, ces actions n'ayant pas d'impact sur la table physique mais uniquement dans cette visualisation
- Générateur de requêtes : créer une requête de type SQL sur la table en cours
- Données : effectuer une tâche de traitement de données sur la table en cours (transposer, fractionner, normaliser etc)
- Description : effectuer une tâche descriptive (listing, rapport, fréquence…)
- Graphique : créer un graphique à partir de la table (histogramme, courbe, nuage de points…)
- Analyser : effectuer une analyse statistique (régression, série chronologique…)
- Exporter la table, le programme ou la log vers un emplacement physique en tant que table SAS, fichier Excel ou texte (txt, csv, tab)
- Envoyer la table en sortie dans un fichier MS Word, Excel ou JMP, ou encore en attachement d'un mail.

2.4.2. La tâche « Tri des données »

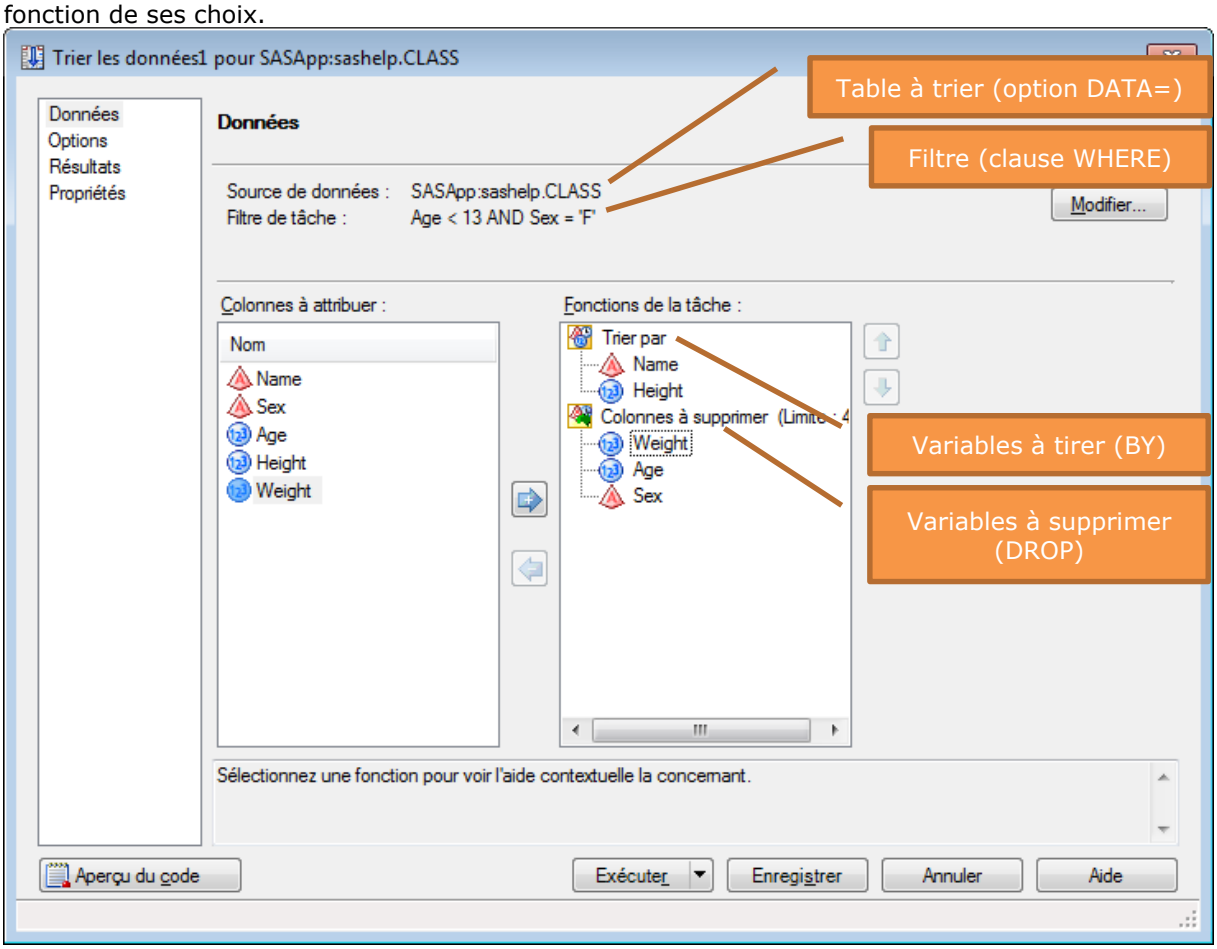
Tout d'abord, l'utilisateur peut sélectionner la table :

- Soit en la glissant dans le flux de processus
- Soit en faisant un clic droit sur la table > Ouvrir.

Puis aller dans le menu Tâches > Données > puis choisir « Trier les données ».



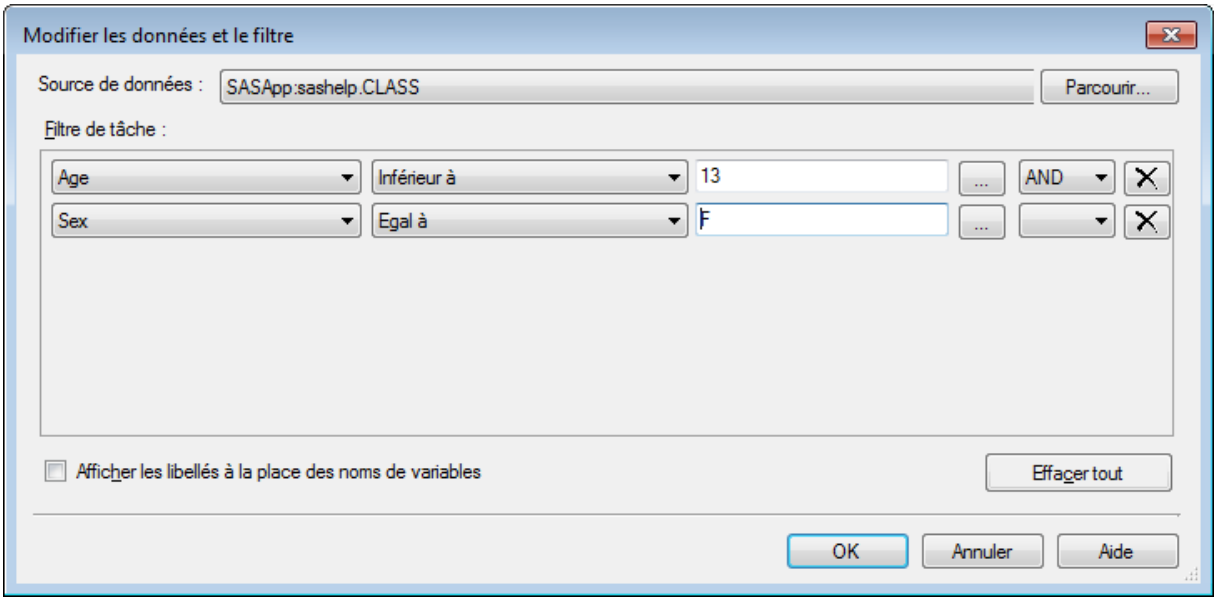
La tâche s'ouvre. Elle est constituée d'une liste de menus que nous allons balayer.
A tout moment, l'utilisateur peut choisir « Aperçu du code » pour savoir quel code est généré en fonction de ses choix.



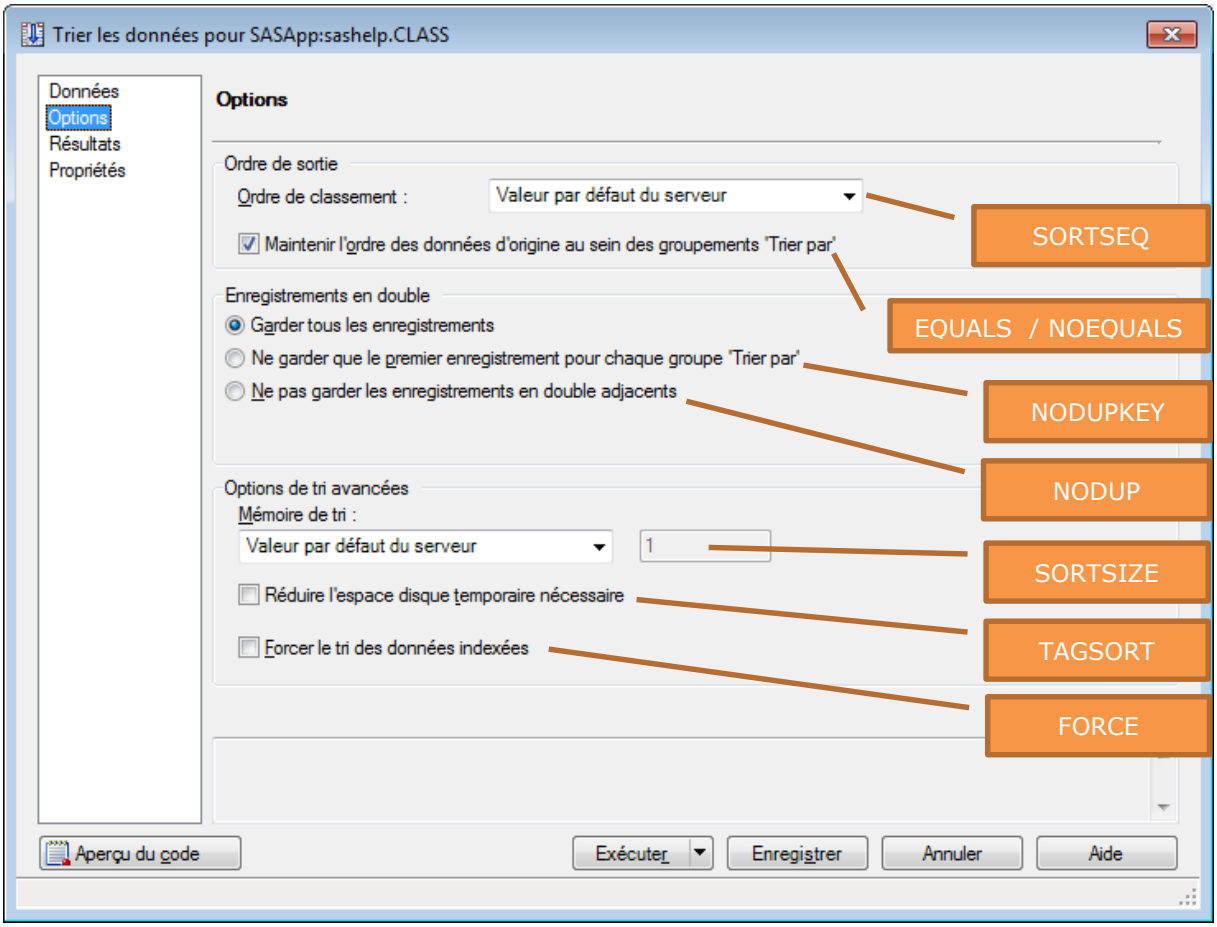
Le menu Données permet de :

- Déterminer la table ou la vue en entrée,
- Définir un filtre (clause WHERE)
- Sélectionner les variables de tri (BY)
- Sélectionner les variables à ne pas garder dans la table en sortie (DROP)

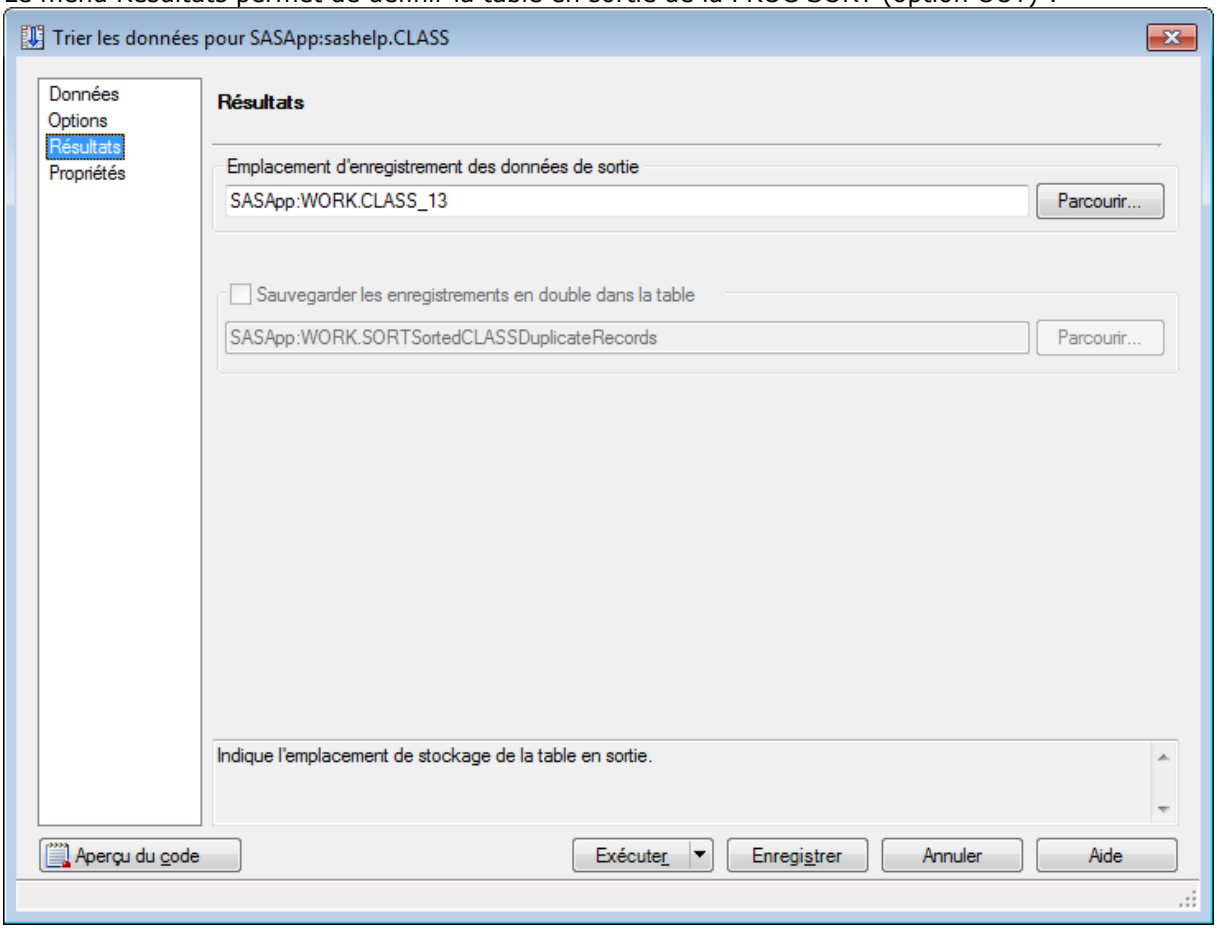
Ils sont modifiables via le menu « Modifier » :



Ensuite, le menu Options permet de définir les options de l'instruction PROC SORT :

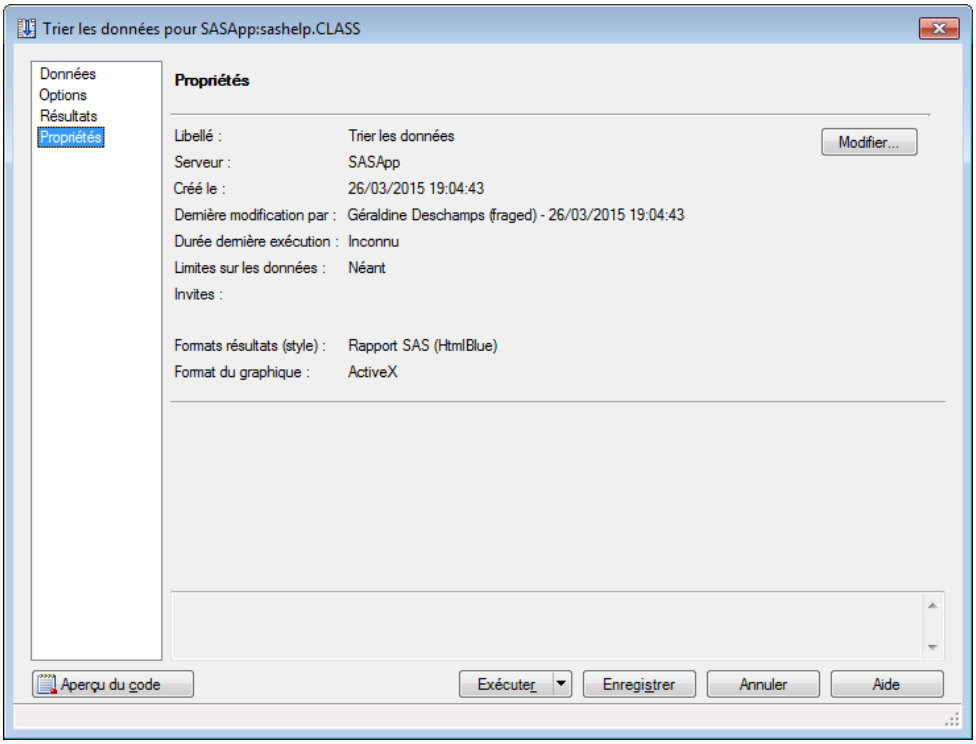


Le menu Résultats permet de définir la table en sortie de la PROC SORT (option OUT) :

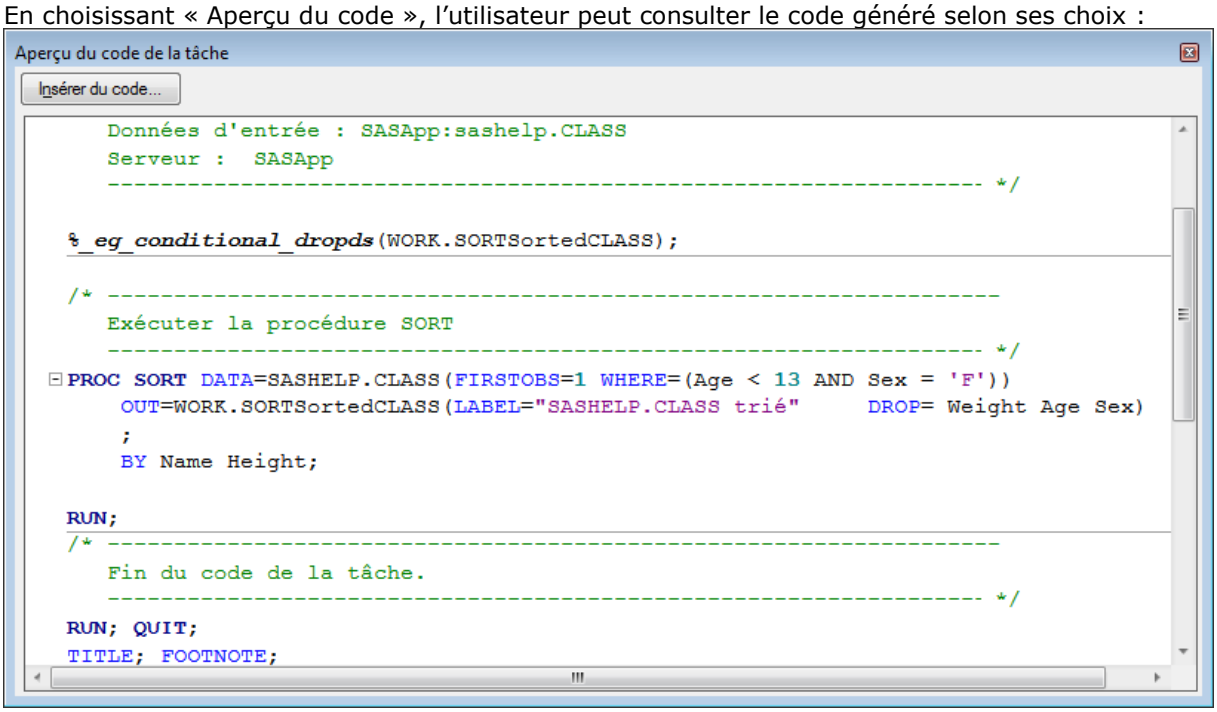


Enfin, le menu Propriétés récapitule les propriétés principales de la tâche. En choisissant Modifier, l'utilisateur va pouvoir, entre autres :

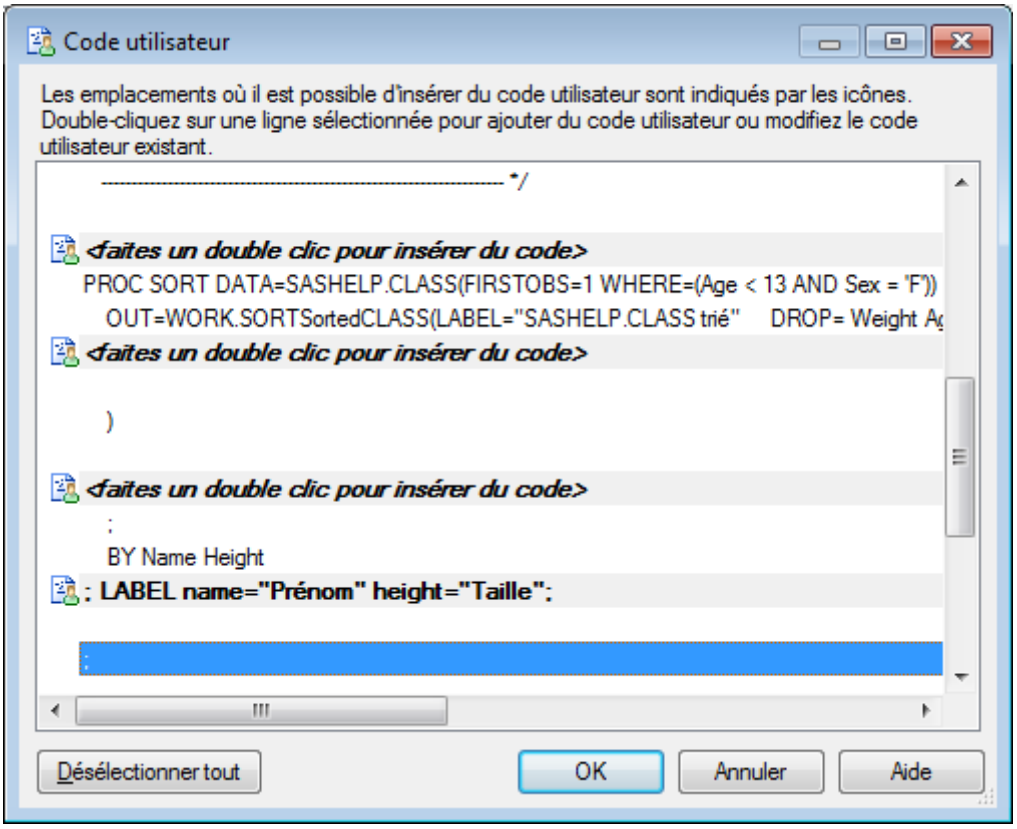
- Modifier le libellé de la tâche dans le projet
- Définir les options FIRSTOBS et NOBS
- Définir une invite



En choisissant « Aperçu du code », l'utilisateur peut consulter le code généré selon ses choix :



Le bouton « Insérer du code » permet d'ajouter des instructions. Dans notre exemple, il reste à ajouter l'instruction LABEL, puis cliquer sur OK :



Une fois que la requête est satisfaisante, l'utilisateur clique sur le bouton « Exécuter ».
En résultat :

- Le journal est accessible.
- La table en sortie est ajoutée au « flux de processus ».
- La table en sortie s'affiche via l'onglet « Données de sortie », et propose des actions à réaliser sur la table (cf. point précédent).

2.5. Depuis SAS Data Integration Studio

Encore une fois, comme pour SAS Studio et SAS Enterprise Guide, l'utilisateur peut trier ses données de plusieurs façons :

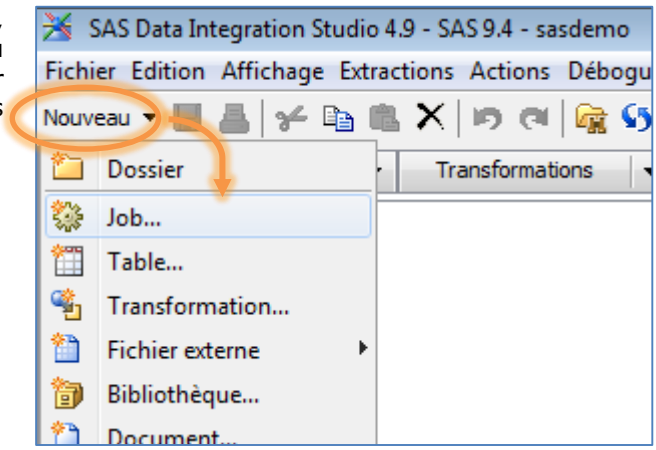
- Il peut écrire manuellement son programme et l'exécuter ;
- Il peut choisir la transformation « Tri » (catégorie « Données »). C'est un assistant pour générer le code de la procédure SORT.
- Il peut choisir la transformation « Extraction » (catégorie « SQL »). C'est un assistant pour générer le code de la procédure SQL.

Nous allons illustrer les 2 dernières méthodes, en reproduisant toujours la même requête :

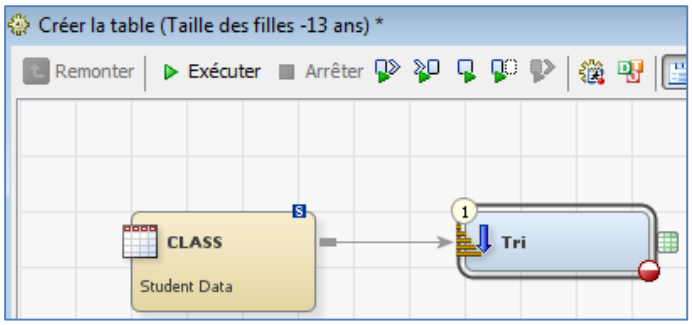
```
PROC SORT DATA=sashelp.class
      OUT=class_13
      (LABEL="Taille des filles -13 ans" KEEP=name height);
  BY name height ;
  WHERE age le 13 and sex='F';
  LABEL name="Prénom" height="Taille";
RUN ;
```

2.5.1. La transformation « Tri »

Il faut d'abord créer un nouveau job, par exemple en sélectionnant Nouveau > Job, lui donner un nom et spécifier un emplacement dans les métadonnées.

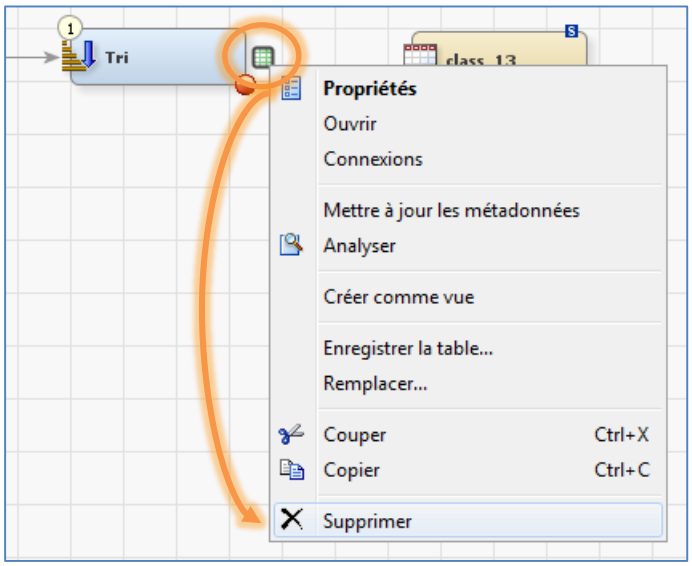


Le job s'ouvre, l'utilisateur glisse ensuite la table en entrée et la transformation Tri. Il les relie ensemble.

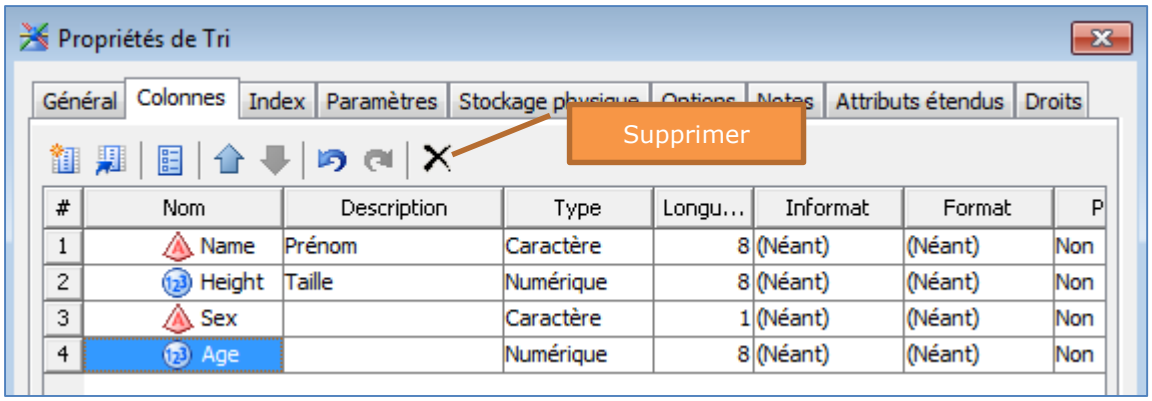


Dans cet exemple, la cible est une table temporaire.
Si l'utilisateur veut enregistrer la table en sortie dans une autre bibliothèque que la work, il doit :

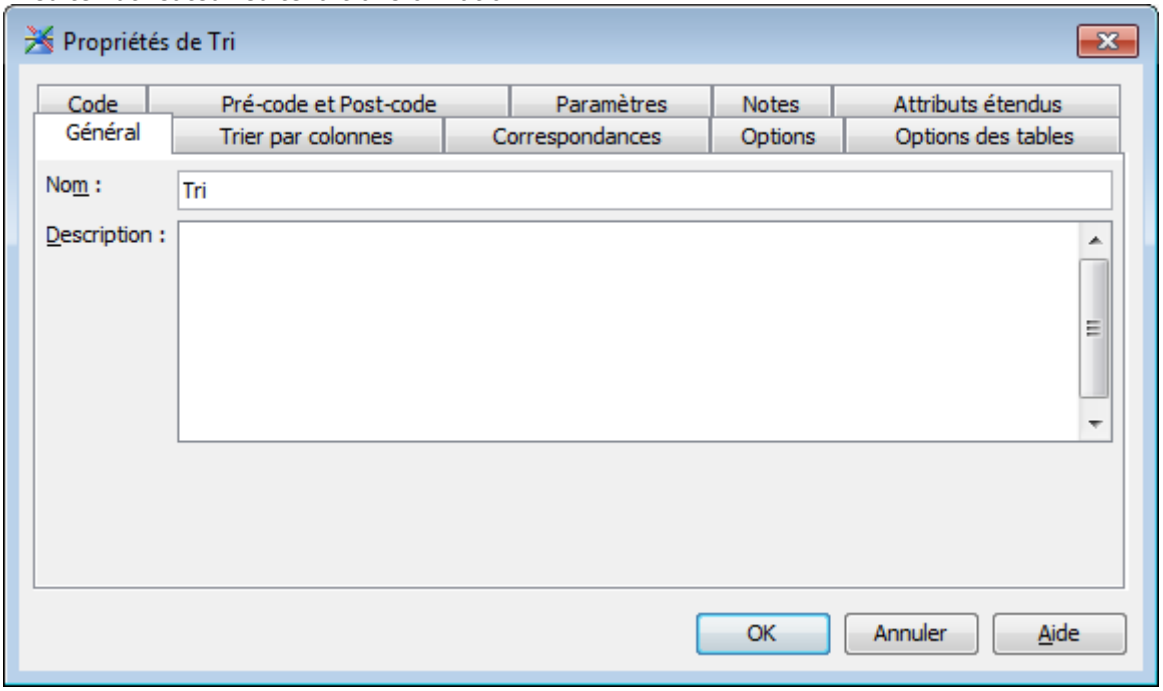
- glisser la table en sortie sur le job,
- supprimer la table temporaire,
- puis relier la transformation « Tri » et la table cible.



Pour notre exemple, il faut supprimer des variables de la table cible, et définir un libellé pour les variables gardées. Pour cela, l'utilisateur édite la table cible et va effectuer ces actions dans l'onglet Colonnes.

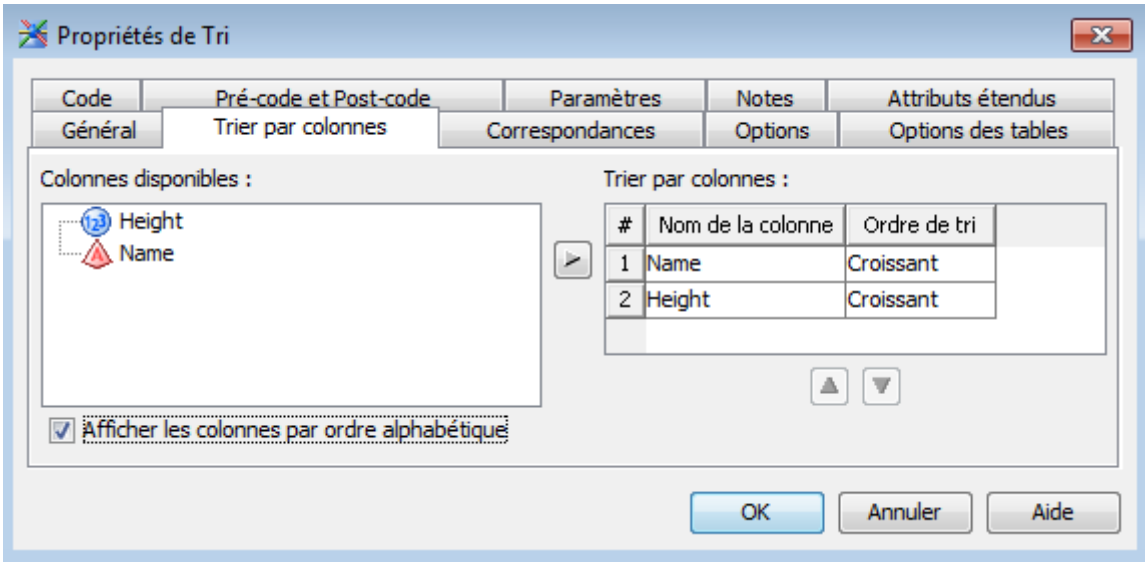


Ensuite l'utilisateur édite la transformation Tri :

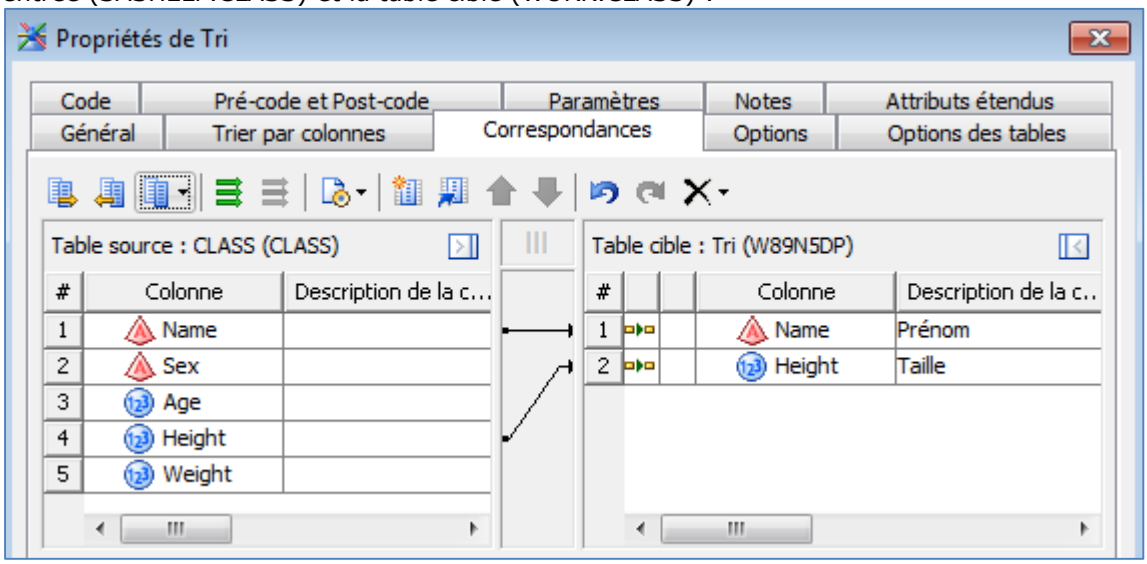


L'onglet Général permet de modifier le nom de la transformation tel qu'il apparaît dans le job, ainsi qu'une description.

L'onglet « Trier par colonnes » permet de définir les variables de tri (instruction BY). La liste des colonnes disponibles correspond aux variables en sortie.

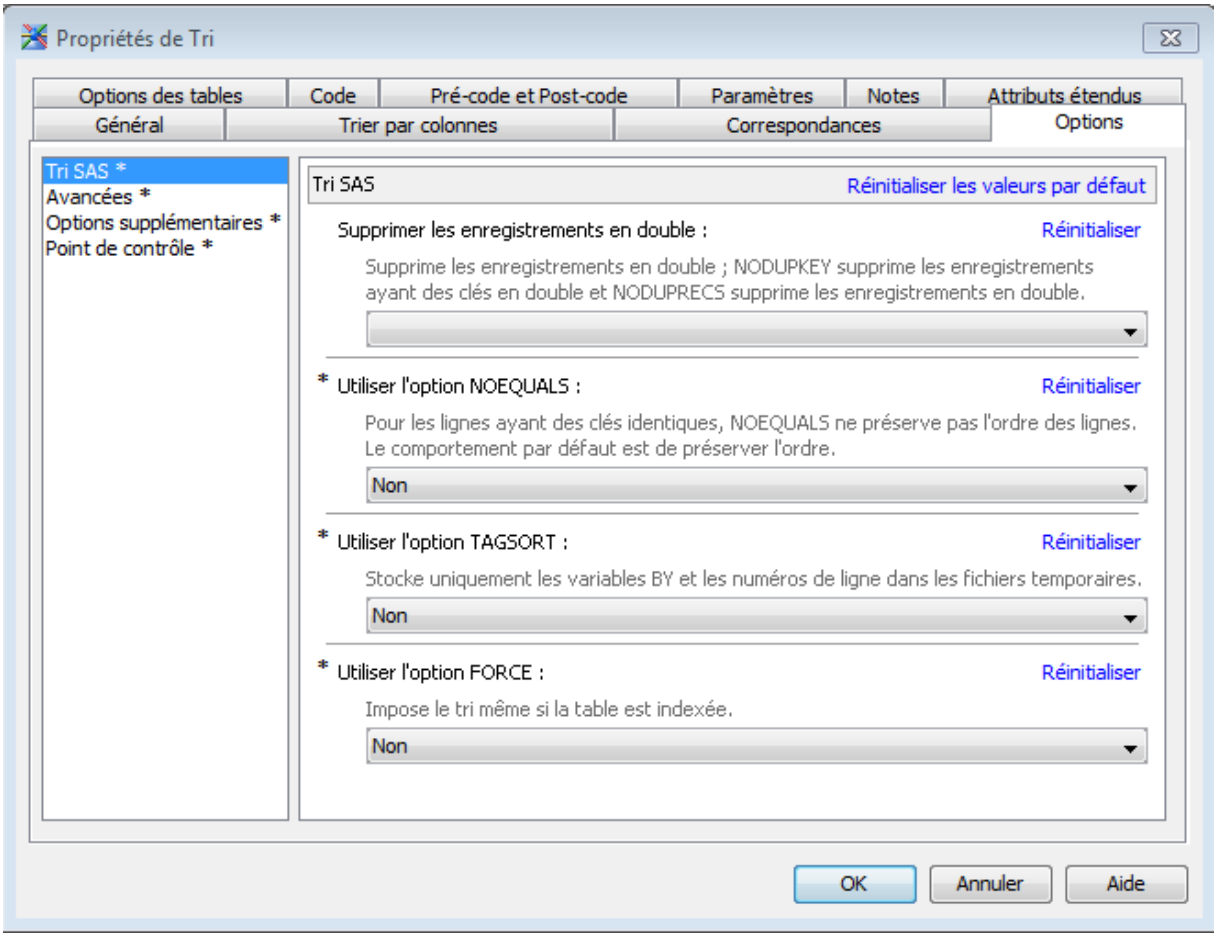


L'onglet « Correspondances » permet de confirmer la correspondance des colonnes entre la table en entrée (SASHELP.CLASS) et la table cible (WORK.CLASS) :

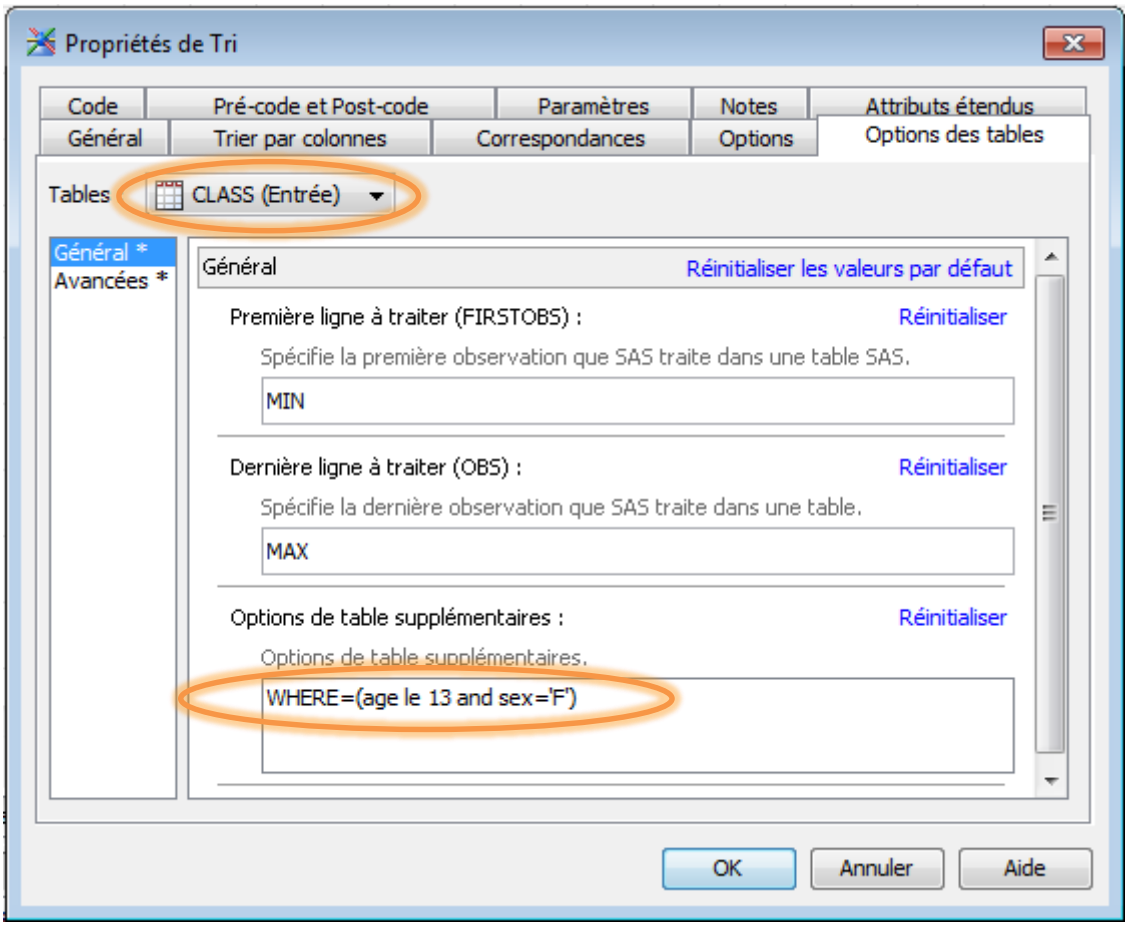


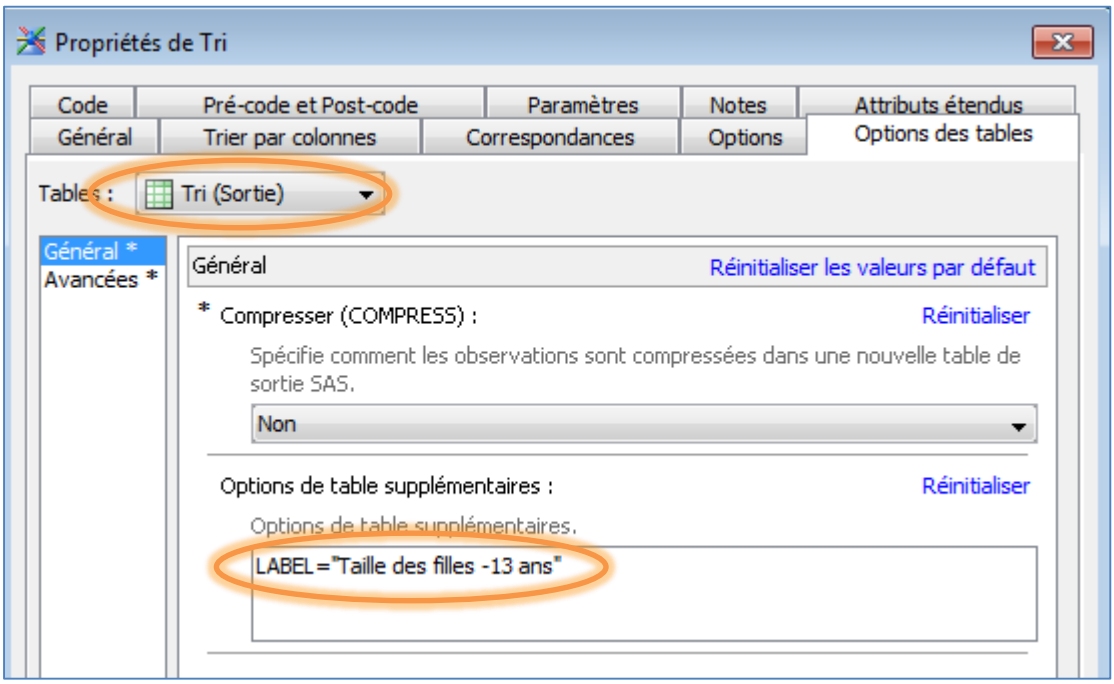
L'onglet « Options » permet de définir des options à différents niveaux :

- Le menu « Tri SAS » permet de déclarer les options NODUPKEY, NODUPRECS, NOEQUALS, TAGSORT, FORCE.
- Le menu « Avancées » permet de déclarer les options SORTSEQ, SORTSIZE, OVERWRITE, mais également un champ libre où l'utilisateur peut noter toute autre option de son choix qui sera ajouté à l'instruction PROC SORT.
- Le menu « Options supplémentaires » ne correspond pas à des options propres à la PROC SORT mais à des actions à ajouter au code, telles que créer la macro &syslast, ajouter des indexes à la table cible, définir les options de l'instruction OPTIONS entre autres.
- Le menu « Points de contrôle » permet de définir les options nécessaires aux modes checkpoint et restart au niveau de la tâche de tri.



L'onglet « Options de tables » permet de définir des options pour la table en entrée et la table en sortie. Dans notre exemple, nous allons pouvoir déclarer le filtre sur la table en entrée, et le libellé de la table en sortie.


Propriétés de Tri
Code
Pré-code et Post-code
Paramètres
Notes
Attributs étendus
Général
Trier par colonnes
Correspondances
Options
Options des tables
Tables
CLASS (Entrée)
Général *
Avancées *
Général
Réinitialiser les valeurs par défaut
Première ligne à traiter (FIRSTOBS) :
Réinitialiser
Spécifie la première observation que SAS traite dans une table SAS.
MIN
Dernière ligne à traiter (OBS) :
Réinitialiser
Spécifie la dernière observation que SAS traite dans une table.
MAX
Options de table supplémentaires :
Réinitialiser
Options de table supplémentaires.
WHERE=(age le 13 and sex='F')
OK
Annuler
Aide


Propriétés de Tri
Code
Pré-code et Post-code
Paramètres
Notes
Attributs étendus
Général
Trier par colonnes
Correspondances
Options
Options des tables
Tables :
Tri (Sortie)
Général *
Avancées *
Général
Réinitialiser les valeurs par défaut
* Compresser (COMPRESS) :
Réinitialiser
Spécifie comment les observations sont compressées dans une nouvelle table de sortie SAS.
Non
Options de table supplémentaires :
Réinitialiser
Options de table supplémentaires.
LABEL="Taille des filles -13 ans"

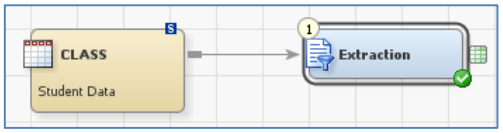
Voici un extrait du code SAS produit par ces choix et correspondant à l'étape du tri :

```
proc sql;
   create view work.W1T56PV as
      select
         Name
            label = 'Prénom',
         Height
            label = 'Taille'
   from &SYSLAST
      (WHERE=(age le 13 and sex='F'))
   ;
quit;

%let SYSLAST = work.W1T56PV;

proc sort data = &SYSLAST
   out = work.W89N5DP
      (LABEL="Taille des filles -13 ans")
      sortsize = 1023 ;
   by
      Name
      Height
   ;
run;
```

2.5.2. La transformation « Extraction »

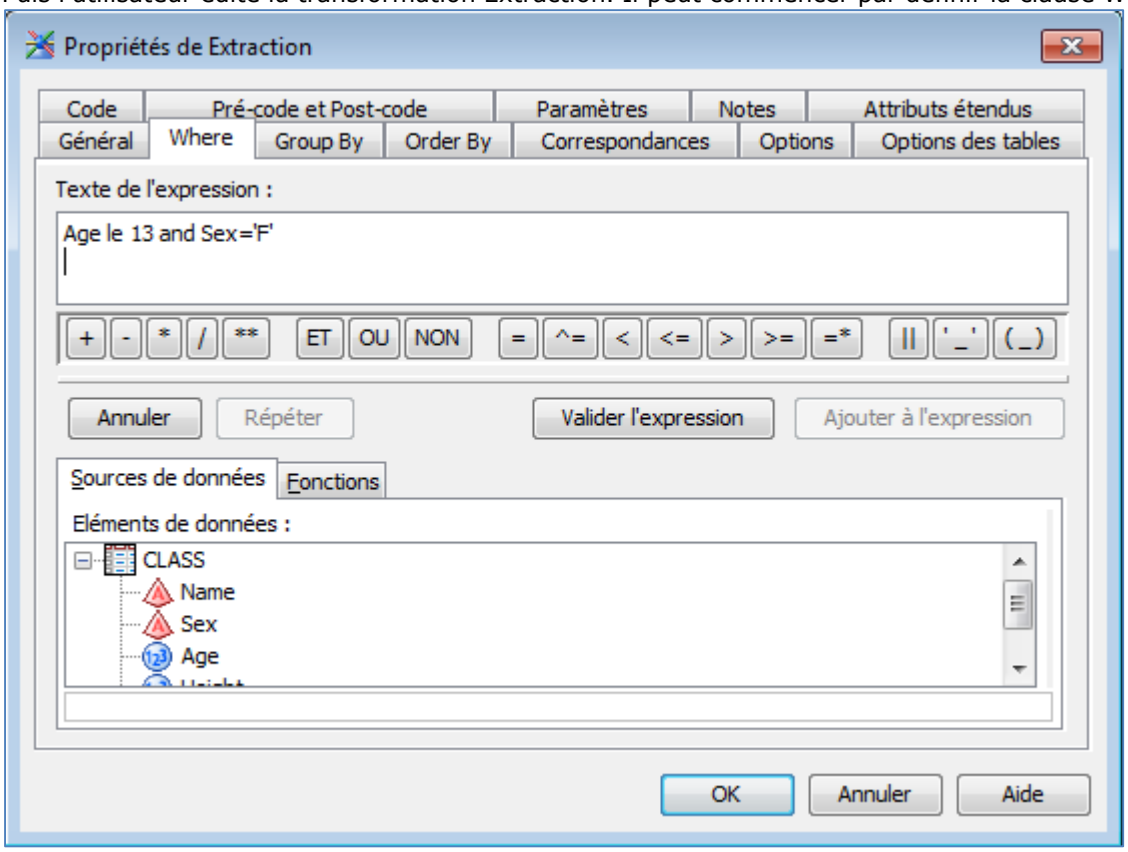
L'utilisateur doit d'abord procéder comme pour la transformation tri :

- Il fait glisser la table en entrée et la transformation puis les relie

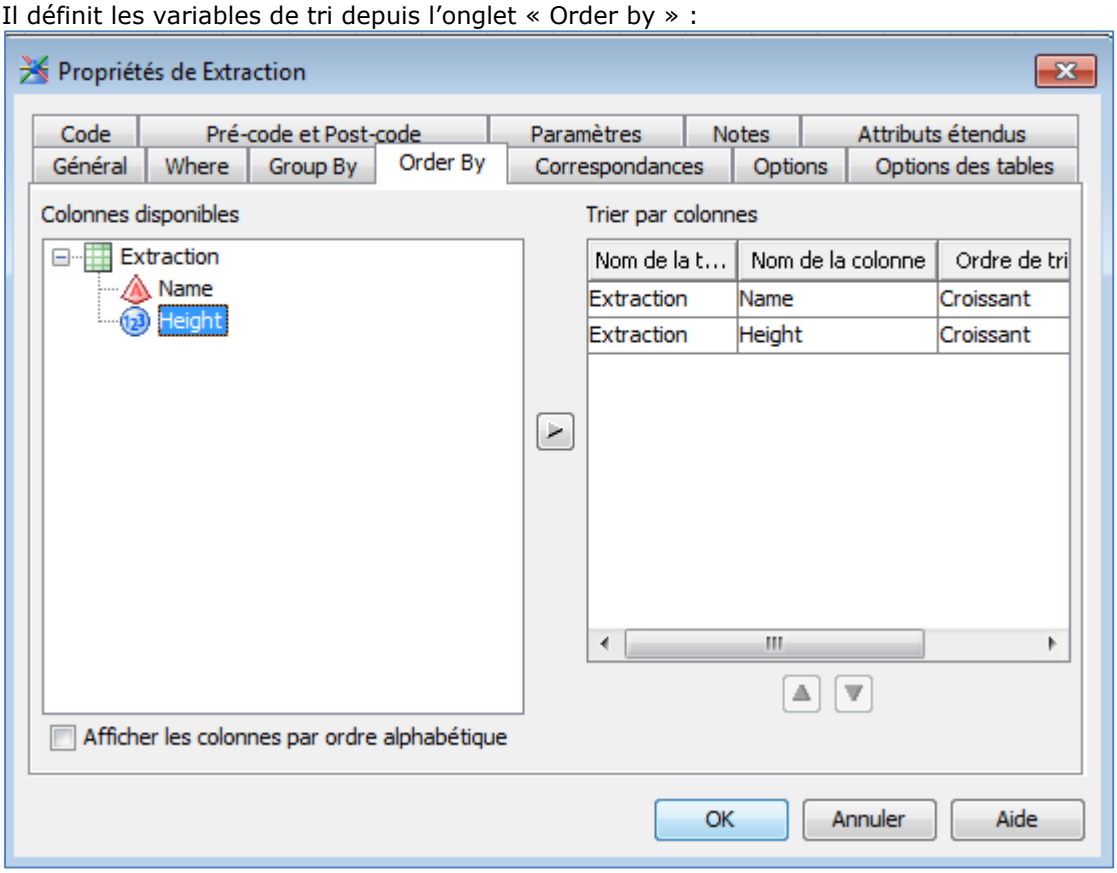


- Il édite la table cible (temporaire ou non) pour sélectionner les colonnes à garder.

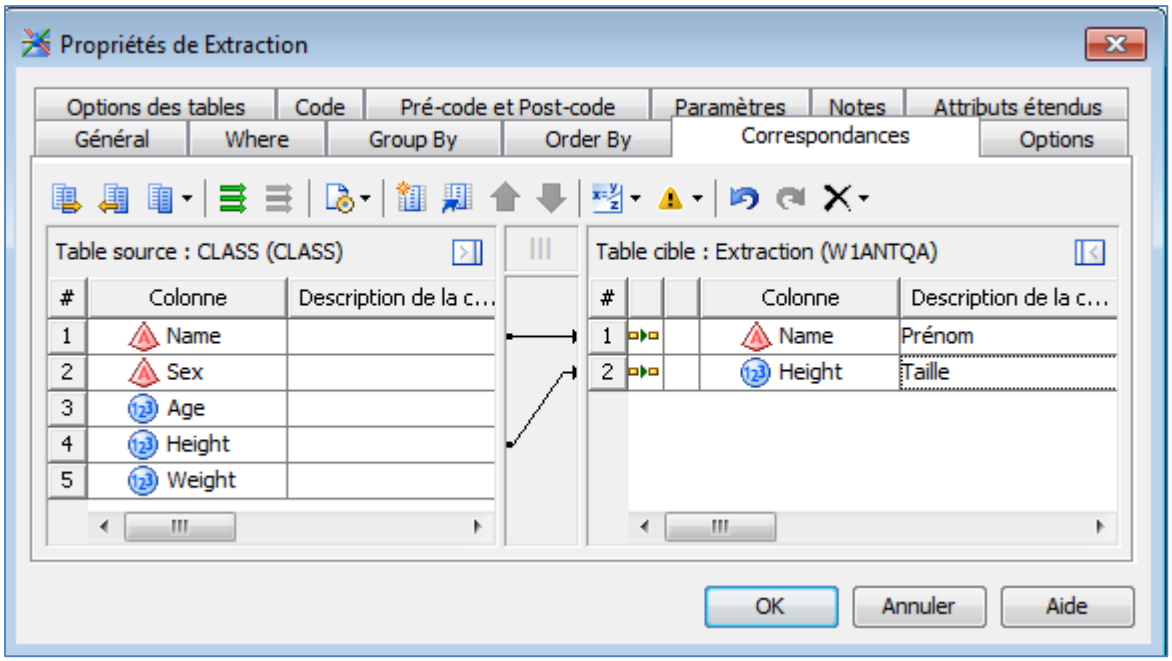
Puis l'utilisateur édite la transformation Extraction. Il peut commencer par définir la clause WHERE :



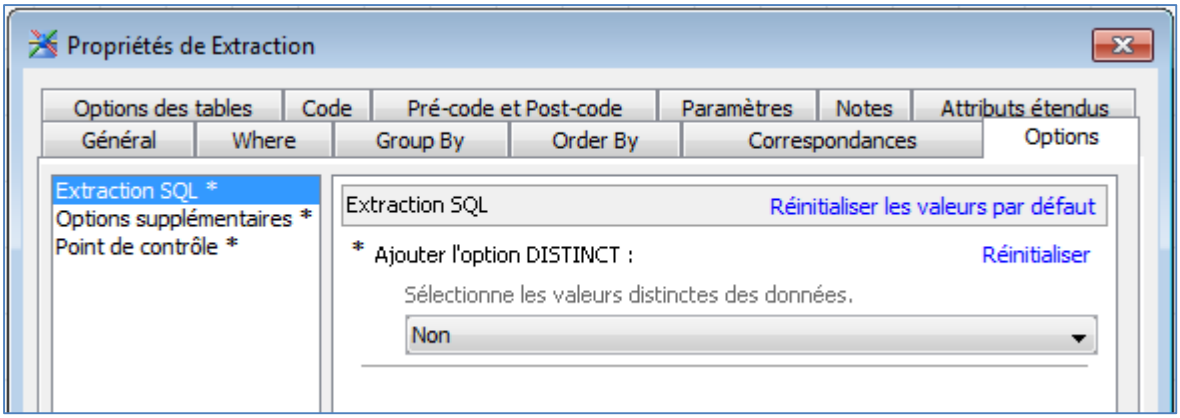
Il définit les variables de tri depuis l'onglet « Order by » :



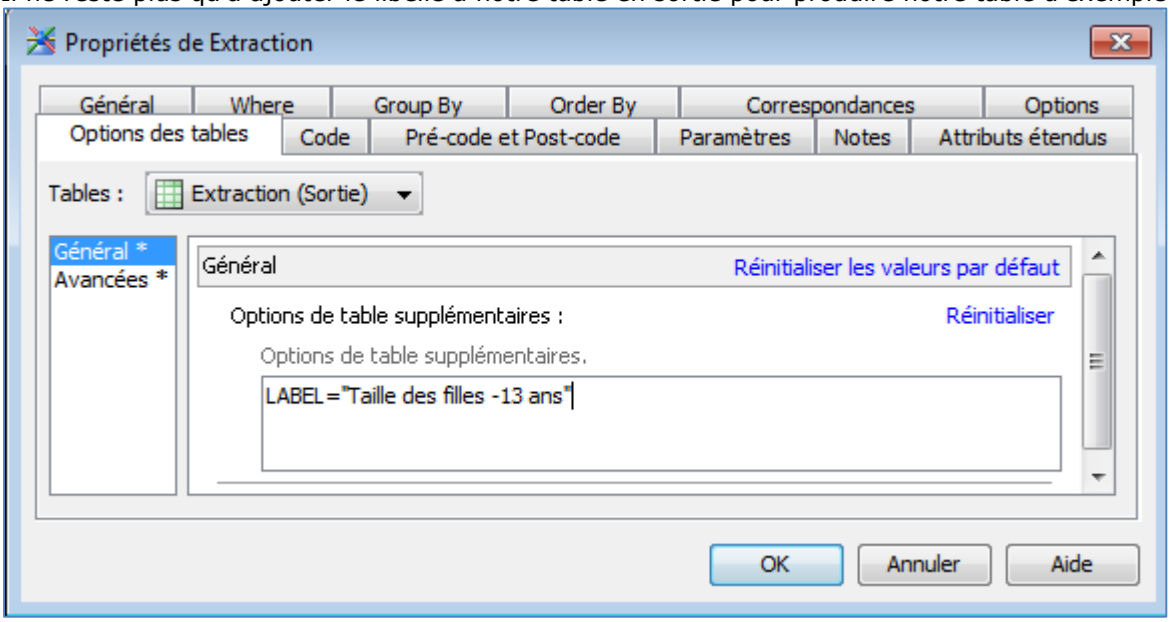
Depuis l'onglet « Correspondances », on vérifie que les variables de la table en entrée sont bien liées aux variables de la table en sortie :



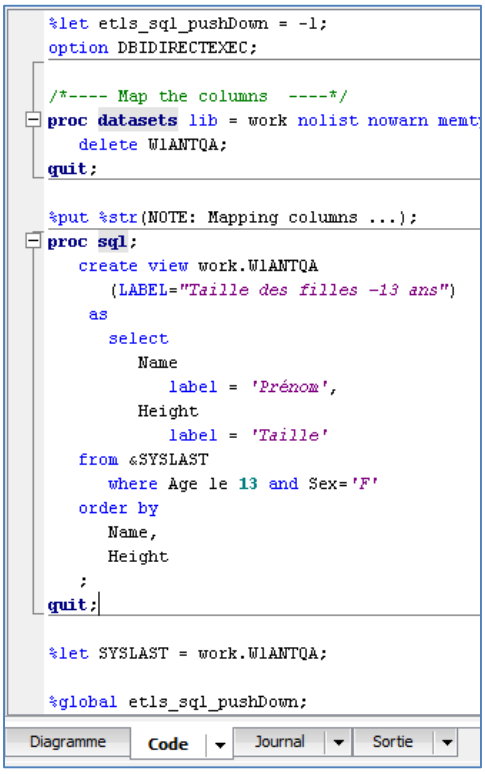
L'onglet « Options » permet d'ajouter l'option DISTINCT à la PROC SQL :



Il ne reste plus qu'à ajouter le libellé à notre table en sortie pour produire notre table d'exemple :



Voici un extrait du code SAS produit par ces choix et correspondant à l'étape du tri :



```
%let etls_sql_pushDown = -1;
option DBIDIRECTEXEC;

/*---- Map the columns  ----*/
proc datasets lib = work nolist nowarn memt
   delete W1ANTQA;
quit;

%put %str(NOTE: Mapping columns ...);
proc sql;
   create view work.W1ANTQA
      (LABEL="Taille des filles -13 ans")
    as
      select
         Name
            label = 'Prénom',
         Height
            label = 'Taille'
   from &SYSLAST
      where Age le 13 and Sex='F'
   order by
      Name,
      Height
   ;
quit;

%let SYSLAST = work.W1ANTQA;

%global etls_sql_pushDown;
```

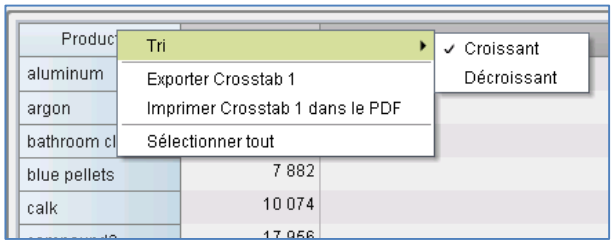
Diagramme | Code | Journal | Sortie

2.6. Depuis SAS Visual Data Builder

Depuis SAS Visual Analytics, on peut distinguer 2 situations concernant les tris :

- Soit on cherche à trier les données utilisées dans une exploration, un rapport, un graphique pour modifier le rendu,
- Soit on cherche à trier une table que l'on créée depuis la SAS Visual Data Builder.

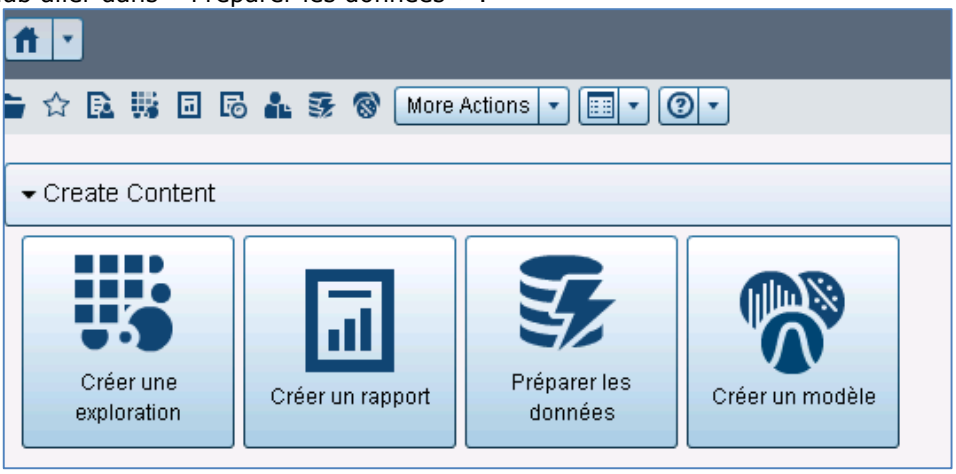
Le sujet de cet article est le tri des données et non pas des sorties, aussi le tri dans SAS® Visual Analytics Designer et SAS Visual Analytics Explorer ne sera pas détaillé. Notons toutefois qu'il est simple à réaliser, puisqu'il suffit de faire un clic-droit sur la variable de catégorie pour pouvoir agir sur le tri.



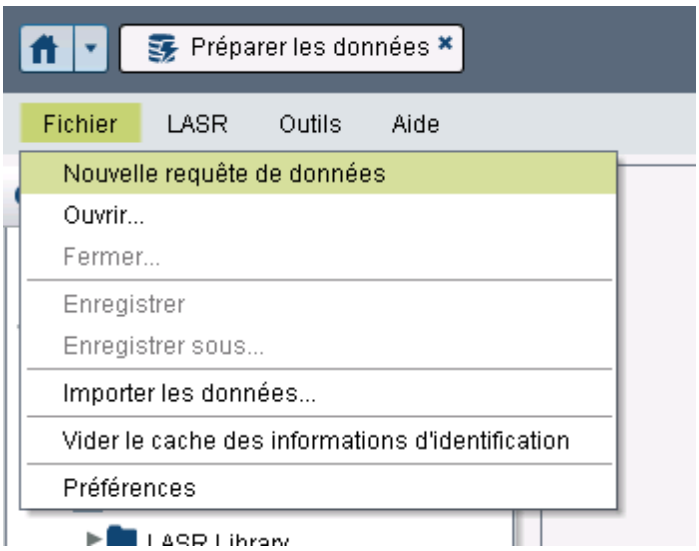
Depuis la version 7.1, l'utilisateur peut créer un tri personnel pour ses rapports : plutôt que de trier par ordre alphabétique, l'utilisateur peut définir son propre tri. La méthode est décrite dans la documentation et également illustrée dans ce post.

Depuis SAS Visual Data Builder, on peut créer une table et la trier. C'est un générateur de procédure SQL : ici ce n'est pas la procédure SORT qui sera illustrée.

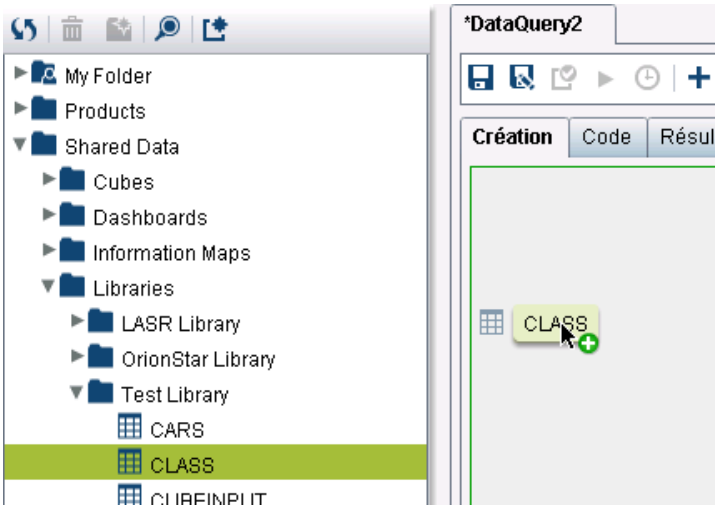
Pour cela, depuis le hub aller dans « Préparer les données » :



puis sélectionner Fichier > Nouvelle requête de données.

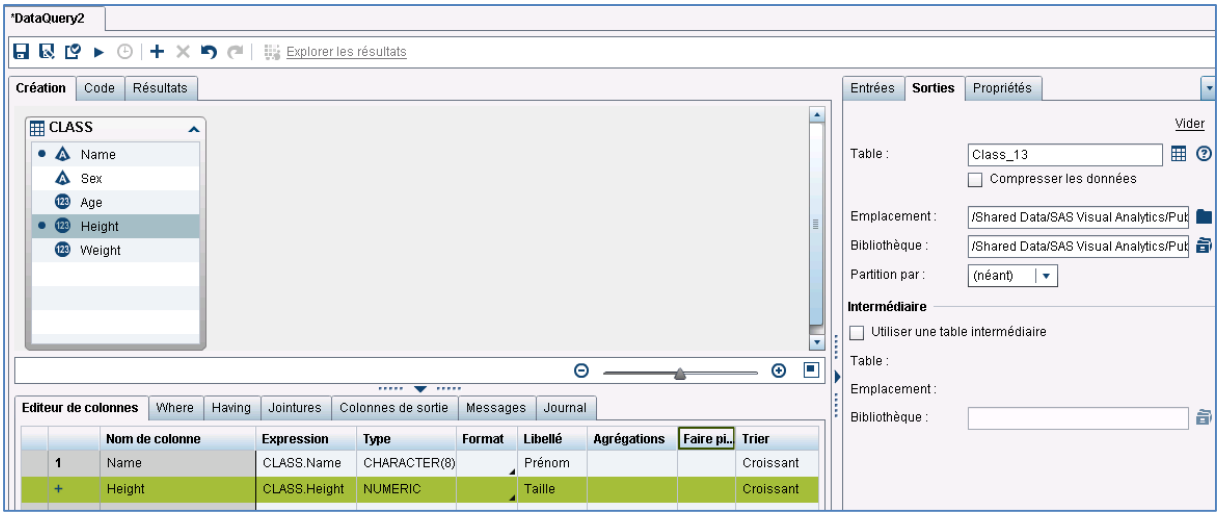


Il faut glisser la/les tables dans la page, puis sélectionner les variables depuis la table source pour qu'elle s'ajoute à la table cible.



Dans cet exemple, nous avons :

- sélectionné les variables en sortie (équivalent du KEEP),
- spécifié pour chacune le tri croissant,
- dans l'onglet Sorties, donné un nom à la nouvelle table et choisi son emplacement dans les métadonnées et sa bibliothèque.



Depuis l'onglet WHERE, nous avons défini le filtre à appliquer :

Ensuite, la requête a été validée via ce bouton : , puis elle a été exécutée : ▶

Les choix réalisés ont produit ce programme :

*DataQuery2

Explorer les résultats

Création | Code | Résultats

Tout le code | Prétraitement | Post-traitement

```
48
49 LIBNAME LASRLIB SASIOLA  TAG=VAPUBLIC  PORT=10031 HOST="sas
50
51 LIBNAME test BASE "c:\data\tables";
52
53 option DBIDIRECTEXEC;
54
55 proc sql noprint;
56  create view TEMP_LASR_VIEW_0 as
57     SELECT
58         CLASS.Name length=8 label='Prénom' AS Name,
59         CLASS.Height length=8 label='Taille' AS Height
60     FROM
61         test.CLASS CLASS
62     WHERE
63         CLASS.Age le 13
64         AND CLASS.Sex='F'
65     ORDER BY
66         Height,
67         Name;
68 quit;
69 /* Drop existing table */
70 %vdb_dt(LASRLIB.Class_13);
71
```

L'utilisateur peut ensuite consulter la table en sortie de manière interactive, et réaliser ces actions :

Atteindre le début ou la fin de la table

Afficher l'outil de recherche

Trier les données

Filtrer les données

Exporter le résultat dans un fichier CSV

*DataQuery2

Explorer les résultats

Création | Code | Résultats

Accéder à la ligne : 1 | En-tête

	Name	Height
1	Alice	56.5
2	Barbara	65.3
3	Jane	59.8
4	Joyce	51.3
5	Louise	56.3

2.7. En code Hash

En code hash, le programmeur aura moins le besoin de trier ses données puisqu'une fusion ne nécessite pas de tri préalable des tables. On peut toutefois trier une table selon la clé, par exemple :

```
data _null_;
  length height 8. name $10. ;
  dcl hash h(dataset: 'sashelp.class (where =(age le 13  and
sex="F"))', ordered: 'a');
  h.DefineKey ('name','height');
  h.DefineData ('name','height');
  h.definedone();
```

```
  h.output(dataset: 'class_13 (LABEL="Taille des filles -13 ans)"');
run;
```

3. IN-DATABASE PROCESSING

Le « In-database » est la capacité de faire exécuter un traitement par le DBMS lorsque le traitement soumis en SAS traite une table du DBMS et que le code est interprétable par celui-ci.

Qu'il soit écrit avec la procédure SORT ou SQL, un tri peut être exécuté par le SGBD de manière tout à fait transparente pour l'utilisateur. Cela permet d'améliorer le temps de traitement du tri et de limiter le transfert des données sur le réseau.

Il y a toutefois quelques points d'attention.
1/ Il faut bien s'assurer que le code de la procédure peut être interprété par le SGBD. Par exemple, l'utilisation d'un format (sans fonction SAS_PUT) dans la clause WHERE peut provoquer le rapatriement des données côté SAS pour effectuer le traitement, et ainsi détériorer les performances.

2/ Le choix de faire un tri SAS ou un tri SGBD dépend de l'option SORTPGM. Elle permet de forcer l'utilisation du tri SAS, ou de laisser le choix à SAS d'utiliser le tri le plus adéquat (SGBD, système ou SAS). C'est un point essentiel pour les performances, mais aussi pour l'ordre des observations qui pourra être différent selon le choix fait. Par exemple, Oracle place les blancs tout à la fin de son tri, tandis que SAS les place en premier.

```
options sortpgm=host ;
```

3/ Lorsque l'on écrit un programme impliquant l'un des modules SAS Access to, il est primordial d'utiliser l'option SASTRACE pour savoir quelles instructions sont exécutées par le SGBD et lesquelles sont exécutées par SAS. Cela permet d'ajuster son code pour une exécution optimale.

```
options sastrace=',,,d' sastraceloc=SASLOG ;
```

Pour en savoir plus sur ce sujet, vous pouvez consulter l'article SAS® in-database : quand le traitement SAS s'exécute dans le SGBD, ainsi que l'aide en ligne In-Database Processing : PROC SORT.

4. EN CAS DE PROBLÈME

4.1. Les problèmes les plus fréquents

4.1.1. Manque d'espace disque

Durant le processus de tri, SAS va créer dans la Work un fichier « utility » qui va grossir tout au long du traitement. On estime qu'il faut 3 à 4 fois la taille de la table en espace libre : par exemple il faut 4 Go d'espace libre pour trier une table d'1 Go. Le document suivant vous explique la méthode pour calculer précisément l'espace nécessaire : Usage Note 15065: How to calculate amount of free work disk space required for PROC SORT.

Attention toutefois : votre estimation de l'espace disque nécessaire au tri peut être erronée si votre table est compressée. Il faut savoir qu'une table compressée est décompressée pour être triée, aussi c'est bien la taille réelle (non compressée) de la table qu'il faut utiliser pour calculer l'espace nécessaire. De plus, il faut noter que l'étape de décompression va elle-même nécessiter de l'espace disque dans la Work, en plus des fichiers temporaires créés pour exécuter le tri.

Lorsqu'il n'y pas assez d'espace disque libre pour réaliser le tri, l'utilisateur reçoit une erreur de ce type :

```
ERROR: Aucun espace disque disponible pour une opération de copie.  Filename =
C:\Users\userid\AppData\Local\Temp\SAS Temporary

ERROR: Espace insuffisant dans le fichier WORK.'SASTMP-000000902'n.UTILITY.
ERROR: Fichier WORK.'SASTMP-000000902'n.UTILITY endommagé. Le traitement E/S
ne s'est pas terminé.
```

4.1.2. Performances

Le sujet est très vaste et ne peut qu'être traité dans un document dédié à cela. Je vous recommande ces deux lectures :

1. Le document de référence en la matière de performances est How to Maintain Happy SAS®9 Users.
2. Si vous recherchez des informations dédiées à la procédure SORT, ce document répondra à toutes vos questions : Getting the Best Performance from V9 Threaded PROC SORT. Il explique notamment le fonctionnement interne de la procédure, et détaille les 3 leviers qui jouent un rôle dans les performances du tri : le système (l'OS, la RAM, le nombre de CPU…), la session SAS (options SORTSIZE, MEMSIZE, UTILLOC etc), et le programme.

4.2. Eléments à transmettre au Support Clients

Si vous êtes client SAS et que vous rencontrez des problèmes liés au tri de vos données, vous pouvez nous soumettre votre problème en écrivant à support@sas.com, en joignant à votre message :

- votre programme,
- la log complète avec l'erreur reçue,
- le résultat d'une PROC CONTENTS sur la table en entrée et la table en sortie,
- le résultat de ceci :

```
proc options group=sort ; run ;
```

5. LIENS UTILES

Les nouveautés de la PROC SORT avec SAS 9.4 sont listées dans la section What's New in Base SAS 9.4 Procedures de la documentation.

La documentation en ligne de la procédure SORT est également indispensable.

Enfin, je vous recommande ces articles:

- Trier des variables selon les valeurs d'une statistique : cet article détaille pas-à-pas la façon dont on peut réordonner les variables d'une table afin que la sortie d'une procédure SG (Statistical Graphics procedures : SGPLOT, SGPANEL and SGSCATTER) soit ordonnée par valeur et non par ordre alphabétique.

- Comment trier les données avec SAS In-Memory Statistics for Hadoop : cet article explique le tri avec le moteur SASIOLA et la PROC IMSTAT.

- Comment trier les lignes d'une matrice avec SAS/IML : ce post illustre l'utilisation de la routine CALL SORT.

- Getting the Best Performance from V9 Threaded PROC SORT : un guide pour améliorer les performances de la PROC SORT.

Géraldine CADE-DESCHAMPS
Consultante Support Clients SAS France