



APPRENTISSAGE DU LANGAGE DE PROGRAMMATION SAS

Gramondo Martina

UT1 FOAD M2 & DU
2018/19





SEMAINE 4:

ETAPE DATA
INSTRUCTION SET
COMPILATION & EXECUTION ETAPE DATA
FILTRE WHERE & IF
CONSTANTES

Gramondo Martina

UT1 FOAD M2 & DU
2018/19



LIRE ET CREER UNE TABLE SAS

INSTRUCTION DATA ET SET

```
DATA SAS-data-set-output;  
SET SAS-data-set-input;  
RUN;
```

L'instruction **DATA** indique la création ou la mise à jour d'une table SAS

L'instruction **SET** indique la table source à partir de laquelle la table en **DATA** va être créée

Dans les deux instructions on indiquera les noms de tables SAS (et des options si on veut).

RAPPEL: les noms des tables SAS sur deux niveaux

- Le nom de la bibliothèque (déjà déclarée), si il n'y aura rien sera la WORK (bibliothèque par default, temporaire)
- Le nom de la table SAS que devra suivre les règles syntaxiques standard des noms dans SAS

RAPPEL: Les noms SAS (variables, tables,...) doivent suivre des règles syntaxiques:

- Ils peuvent compter de 1 à 32 caractères
- Doivent commencer par une lettre ou un _ (souligné du 8)
- Les caractères suivantes (à partir du deuxième) peuvent être des lettres, des chiffres ou des _
- Vous pouvez mélanger la casse des lettres (majuscules et minuscules)
- SAS n'est pas sensible à la casse

LIRE ET CREER UNE TABLE SAS

INSTRUCTION DATA ET SET

```
DATA SAS-data-set-output;  
SET SAS-data-set-input;  
RUN;
```

L'instruction **DATA** peut:

- Modifier une table déjà existant si le nom de la table existe déjà
- Créer une nouvelle table si le nom de table ne existe pas encore

L'instruction **SET** :

- Permet de lire toutes les observations et toutes les variables de la table de l'instruction
- Les observations sont lues dans une façon séquentielle (une ligne après l'autre)
- La table dans l'instruction SET doit exister

NB: dans l'instruction DATA on peut écrire un ou plusieurs noms de table SAS et on pourra ajouter des options.

NB: dans l'instruction SET on peut écrire un ou plusieurs noms de tables SAS (concaténation) et on pourra ajouter des options.

ETAPE DATA

COPIER LES DONNÉES D'UNE TABLE A L'AUTRE

Etape DATA
DATA

Nom de la table crée **WORK.CLASS**

Nom de la table source **sashelp.class**

Instruction
SET

```
data class;  
set sashelp.class;  
run;
```

Exemple

```
DATA SAS-data-set-output;  
SET SAS-data-set-input;  
RUN;
```

Code standard

Fin de l'étape
RUN

ETAPE DATA RESULTAT

```
73      data class;  
74      set sashelp.class;  
75      run;
```

NOTE: There were 19 observations read from the data set SASHELP.CLASS.
NOTE: The data set WORK.CLASS has 19 observations and 5 variables.
NOTE: DATA statement used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

Fenêtre
Journal

CODE		JOURNAL	RESULTATS	DONNEES EN SORTIE
Table :		WORK.CLASS	Afficher :	Libellés de la colonne
Colonnes		Lignes totales : 19 Colonnes totales : 5		
<input checked="" type="checkbox"/>	Sélectionner tout			
<input checked="" type="checkbox"/>	Name			
<input checked="" type="checkbox"/>	Sex			
<input checked="" type="checkbox"/>	Age			
<input checked="" type="checkbox"/>	Height			
<input checked="" type="checkbox"/>	Weight			
Propriété		Valeur		
Libellé				
Nom				
Longueur				
Type				
Format				

	Name	Sex	Age	Height
1	Alfred	M	14	69
2	Alice	F	13	56.5
3	Barbara	F	13	65.3
4	Carol	F	14	62.8
5	Henry	M	14	63.5
6	James	M	12	57.3
7	Jane	F	12	59.8
8	Janet	F	15	62.5
9	Jeffrey	M	13	62.5
10	John	M	12	59
11	Joyce	F	11	51.3
12	Judy	F	14	64.3
13	Louise	F	12	56.3
14	Mary	F	15	66.5
15	Philip	M	16	72

Fenêtre
Données en
Sortie

Si on veut contrôler le bloque
descripteur de notre nouvelle table:
`proc contents data=class;`
`run;`

Si on veut afficher un rapport de notre
nouvelle table:
`proc print data=class;`
`run;`

COMPILATION & EXÉCUTION

COMMENT TRAVAILLE SAS EN ARRIERE PLAN

Quand nous allons appuyer sur  notre programme est envoyé en RUN

Un étape DATA est traité principalement en 3 phases:

- La phase de **Pré-Compilation** dans laquelle on aura le contrôle des token et le **contrôle syntaxique** du code
- La phase de **Compilation pour gérer la structure**
- La phase de **Exécution pour gérer les données**

NB: L'utilisation du terme Compilation est impropre car SAS est un programme pré compilé

COMPILATION

EXPLICATION

La phase de **Compilation**:

- Elle est faite après la phase de Pré-Compilation
- Elle est faite une fois seulement par étape data
- Elle gère la création du Program Data Vector (PDV)

Le Program Data Vector (PDV) est un buffer (un espace disque temporaire) contenant toutes les variables rencontrées dans votre étape DATA.

Les variables peuvent être:

- Une anciennes variables, issue de la table de l'instruction SET, MERGE, UPDATE, ...
- Une nouvelles variables crée dans la l'étape DATA

Le PDV donne une case mémoire à chaque variable rencontré dans la lecture de votre programme.

La case mémoire dans le PDV est crée à la première lecture de la variable, il affectera à la variable les attributs obligatoires (nom, type, longueur) et facultatifs (format et label).

La structure du PDV ne sera plus changé et constituera le bloque descripteur de la table en sortie (sauf instructions `_NULL_`, `KEEP`, `DROP`, ...)

COMPILATION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
run;
```

SAS crée un espace dans la **WORK** pour y stocker la future table
WORK.CLASS

COMPILATION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
run;
```

SAS rencontre des variables via l'instruction SET

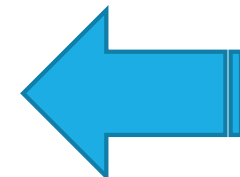
SAS récupère le bloque descripteur des variables de la table **SASHELP.CLASS**
c'est la première fois que il va lire ces variables pendant la compilation de cette étape data et donc il va les créer dans le PDV

SAS va copier toutes les variables avec leur attributs dans l'ordre de la table source.

Bloque descripteur des variables de SASHELP.CLASS

Liste alphabétique des variables et des attributs

#	Variable	Type	Long.
3	Age	Num.	8
4	Height	Num.	8
1	Name	Texte	8
2	Sex	Texte	1
5	Weight	Num.	8



Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT
\$ 8	\$ 1	N 8	N 8	N 8

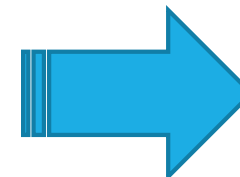
COMPILATION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
run;
```

RUN indique la fin de l'étape et donc la fin de la compilation
SAS va copier le PDV dans l'espace **WORK.CLASS**, sera son bloque descripteur des variables
SAS peut commencer l'exécution

Program Data Vector				
NAME	SEX	AGE	HEIGHT	WEIGHT
\$ 8	\$ 1	N 8	N 8	N 8



Blocue descripteur des variables de WORK.CLASS		
NOM	TYPE	LONG
NAME	\$	8
SEX	\$	1
AGE	N	8
HEIGHT	N	8
WEIGHT	N	8

EXECUTION

EXPLICATION

La phase d'Exécution:

- Elle est faite après la phase de Compilation
- Elle est faite pour chaque ligne en entrée
- Elle crée automatiquement deux variables pour suivre l'exécution (_N_ et _ERROR_)
- Elle est conditionnée par les valeurs des variables pour choisir les instructions à exécuter (IF, ELSE, SELECT,...)

L'exécution d'une étape DATA dans SAS est une boucle que va être faite autant des fois que il y a des lignes en entrée (SET, OBS, FIRSTOBS, ...)

Les instructions que vont définir le nombre des interactions de la boucle d'exécution:

- SET indique la table en entrée et, par default, sera donc le nombre des observations en entrées que vont indiquer le nombre des fois que on fera l'exécution
- WHERE filtre en entrés on aura autant des exécutions que des lignes de notre table SET que vont vérifier la condition
- ... et des autres instructions que on verra plus tard (OBS, FIRSTOBS, MERGE, ...)

EXECUTION

EXPLICATION

Première exécution lecture instruction DATA:

- Deux variables automatiques sont créées et seront automatiquement supprimées à la fin de l'exécution:
 - `_N_=1` est une variable numérique, elle est le numéro de l'exécution, le compteur de la boucle
 - `_ERROR_=0` est une variable booléenne long 1 bit, elle prend la valeur 1 quand il y a présence d'une erreur pendant cette exécution, si il n'y a pas d'erreur elle restera à 0
- Le PDV est initialisé à missing pour toutes les variables (sauf pour `_N_`, `_ERROR_` et les variables RETAIN)

Pour les autres exécutions, la lecture de l'instruction DATA:

- `_N_` aura un incrément de 1 pour chaque exécution
- Le PDV initialisera que les nouvelles variables (variables créées dans l'étape) à missing, alors que les autres vont garder leur valeur qui sera écrasé par la lecture de une nouvelle ligne dans le fichier source avec l'instruction SET (l'instruction RETAIN change cet comportement mais on le verra pas tout de suite)

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
run;
```

← SAS crée les variables `_N_` et `_ERROR_`
SAS initialise toutes les variables du PDV à missing

Program Data Vector						
NAME \$ 8	SEX \$ 1	AGE N 8	HEIGHT N 8	WEIGHT N 8	_N_ N 8	_ERROR_ N 1
		.	.	.	1	0

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
run;
```

SET copie une ligne de la table en input
SASHELP.CLASS dans le PDV

Zone des données de
SASHELP.CLASS

Obs.	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	N 8	N 1
Alfred	M	14	69.0	112.5	1	0

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION



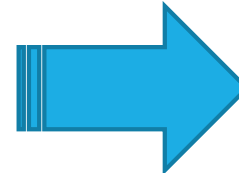
```
data class;  
set sashelp.class;  
run;
```

Fin exécution `_N_=1`

Sortie implicite, le PDV écrit la première observation dans la table **WORK.CLASS**
(Les variables `_N_` et `_ERROR_` ne sont pas copiées)

Retour implicite, on recommence une autre boucle, donc `_N_=2`

Program Data Vector						
NAME	SEX	AGE	HEIGHT	WEIGHT	<code>_N_</code>	<code>_ERROR_</code>
\$ 8	\$ 1	N 8	N 8	N 8	N 8	N 1
Alfred	M	14	69.0	112.5	1	0



Zone de données de WORK.CLASS				
Alfred	M	14	69.0	112.5

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
run;
```

← Début exécution `_N_=2`

`_ERROR_` est réaffecté à 0

A partir de `_N_=2` SAS initialise dans le PDV que les nouvelles variables (les variables créées dans l'étape DATA) les autres gardent les valeurs de l'exécution précédente, pour le moment

Program Data Vector

NAME \$ 8	SEX \$ 1	AGE N 8	HEIGHT N 8	WEIGHT N 8	_N_ N 8	_ERROR_ N 1
Alfred	M	14	69.0	112.5	2	0

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
run;
```

SET permet de écraser le valeurs des
(anciennes) variables de l'exécution précédente
pour copier une autre ligne de la table en input
SASHELP.CLASS

Zone des données de **WORK.CLASS**

Obs.	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	N 8	N 1
Alice	F	13	56.5	84.0	2	0

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION



```
data class;  
set sashelp.class;  
run;
```

Fin exécution `_N_=2`

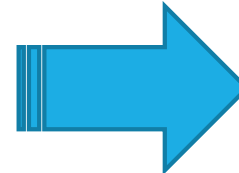
Sortie implicite, le PDV écrit la première observation dans la table **WORK.CLASS**

Les variables `_N_` et `_ERROR_` ne sont pas copiées)

Retour implicite, on recommence une autre boucle `_N_=3`

... et on va continuer jusqu'à la fin de la table **SASHELP.CLASS**

Program Data Vector						
NAME	SEX	AGE	HEIGHT	WEIGHT	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	N 8	N 1
Alice	F	13	56.5	84.0	2	0



Zone de données de WORK.CLASS				
Alfred	M	14	69.0	112.5
Alice	F	13	56.5	84.0

ETAPE DATA RESULTAT

```
73      data class;  
74      set sashelp.class;  
75      run;
```

NOTE: There were 19 observations read from the data set SASHELP.CLASS.
NOTE: The data set WORK.CLASS has 19 observations and 5 variables.
NOTE: DATA statement used (Total process time):
 real time 0.01 seconds
 cpu time 0.01 seconds

Fenêtre
Journal

CODE

JOURNAL

RESULTATS

DONNEES EN SORTIE

Table :

WORK.CLASS

Afficher :

Libellés de la colonne

Filtrer : (néant)

Colonnes

Sélectionner tout

Name

Sex

Age

Height

Weight

Propriété

Valeur

Libellé

Nom

Longueur

Type

Format

Lignes totales : 19

Colonnes totales : 5

Lignes 1-19

	Name	Sex	Age	Height
1	Alfred	M	14	69
2	Alice	F	13	56.5
3	Barbara	F	13	65.3
4	Carol	F	14	62.8
5	Henry	M	14	63.5
6	James	M	12	57.3
7	Jane	F	12	59.8
8	Janet	F	15	62.5
9	Jeffrey	M	13	62.5
10	John	M	12	59
11	Joyce	F	11	51.3
12	Judy	F	14	64.3
13	Louise	F	12	56.3
14	Mary	F	15	66.5
15	Philip	M	16	72

Fenêtre
Données en
Sortie

Si on veut contrôler le bloque
descripteur de notre nouvelle table:
`proc contents data=class;`
`run;`

Si on veut afficher un rapport de notre
nouvelle table:
`proc print data=class;`
`run;`

SELECTION OBSERVATIONS

INSTRUCTION WHERE

```
DATA SAS-data-set-output;  
SET SAS-data-set-input;  
WHERE operande operateur;  
RUN;
```

L'instruction **WHERE** permet de définir des règles pour sélectionner les observations admise en exécution

Dans l'instruction **WHERE** nous allons utiliser des expression:

- Si l'expression est vérifiée l'observation pourra rentrer dans la boucle de l'exécution (_N_)
- Si l'expression n'est pas vérifiée l'observation ne rentre pas dans la boucle de l'exécution, l'observation suivante sera vérifiée

L'instruction **WHERE** est un filtre en entré, donc la phase de exécution sera faite autant des fois que il y aura des observations que vont vérifier l'expression (et non pas autant des fois que il y a de ligne dans la table en entré)

La variable utilisée dans le **WHERE** doit être une ancienne variable (càd variable héritée d'un SET,...)

Une seule instruction **WHERE** est accepté par étape

Une instruction **WHERE** peut être écrire dans un ETAPE DATA et dans la plupart des étapes PROC

SELECTION OBSERVATIONS

INSTRUCTION WHERE

```
DATA SAS-data-set-output;  
SET SAS-data-set-input;  
WHERE operande opérateur;  
RUN;
```

L'expression est composé d'un **opérande**:

- Constante caractère, numérique, date, heure, date&heure
- Variable numérique, caractère et d'un opérateur

Et un ou des **opérateurs**:

- Symboles qui représentent une comparaison, calcul, ou un opérateur logique
- Opérateur réservé au WHERE
- Fonctions SAS

SELECTION OBSERVATIONS CONSTANTES

Les **constantes** sont des **valeurs fixes**:

- Les valeurs de type caractère sont placées entre guillemets et sont sensibles à la casse
- Les valeurs numériques n'utilisent pas de guillemets ou de caractères spéciaux

La **constante date** est une **façon de écrire une date** pour indiquer à SAS que il faut compter le nombre des jours entre la date de référence (01 Janvier 1960) et la date de la constante date.

Il faudra l'écrire dans la façon suivante: '25DEC2018'd

- Toute entre accolades simples ou doubles (' ' ou " ")
- Le jour sur 1 ou 2 (ex. 01 ou 1)
- Le mois en 3 lettre en anglais, pas sensible à la casse (ex. DEC ou dec pour December)
- L'année sur 2 ou 4 (ex. 2018 ou 18)
- La lettre d pas sensible à la casse (d ou D)

SELECTION OBSERVATIONS

EXEMPLES

```
data class;  
set sashelp.class;  
where age >=15;  
run;
```

Variable **AGE** ancienne variable car héritée de **SASHELP.CLASS**

Constante numérique

```
data class;  
set sashelp.class;  
where Sex='M';  
run;
```

Variable **SEX** ancienne variable car héritée de **SASHELP.CLASS**

Constante caractère

```
data class;  
set sashelp.class;  
where today()='25DEC2018'd;  
run;
```

fonction **TODAY ()** récupérer la date système pas de création de variable, toute est fait à la volée

Constante date

SELECTION OBSERVATIONS

OPERATEURS DE COMPARAISON

Les operateurs de comparaisons sont:

SYMBOLE	MNEMONIQUE	DEFINITION
=	EQ	Egal à
\neq $\neg =$ $\sim =$	NE	Diffèrent de
>	GT	Strictement supérieur à
<	LT	Strictement inférieur à
>=	GE	Supérieur ou égale à
<=	LE	Inférieur ou égal à
	IN	Dans une liste

NB: les operateurs de la liste pourront être utilisés aussi dans des autres instructions de comparaison (IF, SELECT WHEN,...)

SELECTION DES OBSERVATIONS

EXEMPLES

```
data class;  
set sashelp.class;  
where age GE 15;  
run;
```

Superieur eguale >=

```
data class;  
set sashelp.class;  
where Sex EQ 'M';  
run;
```

Egual =

```
data class;  
set sashelp.class;  
where name in ('Alice', 'Barbara', 'Carol');  
run;
```

Dans une liste

Si il y a des caractères il faut mettre les ''

SELECTION OBSERVATIONS

CONDICTIONS ENTRE OPERATEURS

Nous avons le droit de utiliser seulement une instruction **WHERE** par étape DATA ou PROC

Pour contrôler plusieurs valeurs ou variables différentes faudra utiliser les **opérateurs logiques** :

SYMBOLE	MNEMONIQUE	DEFINITION
&	AND	Les deux conditions
	OR	Au moins une condition
^ ¬ ~	NOT	Negation

Avec des parenthèses on peut mettre une **priorité** (ex: (age <10 OU age >15) AND (Sex='F'))

SELECTION OBSERVATIONS

EXEMPLES

```
data class;  
set sashelp.class;  
where age > 10 AND age <15;  
run;
```

Age comprise entre 11 et 14

```
data class;  
set sashelp.class;  
where Height > 65 OR Weight >100;  
run;
```

La taille ou le poids doivent suivre les limites

```
data class;  
set sashelp.class;  
where name='Alice' OR name = 'Barbara OR  
name= 'Carol';  
run;
```

La traduction de l'operateur IN
Where name in ('Alice', 'Barbara',
'Carol');

SELECTION OBSERVATIONS

QUESTION

Sachant que:

SEX est variable caractère, WEIGHT variable numérique, DATE_NAISSANCE variable numérique

Je veux garder dans ma table en résultat seulement les filles avec un poids inférieur à 50 ou supérieur à 100 nées après le 01/01/2005.

Choisir la bonne proposition de l'instruction WHERE :

- A `Where sex=F and (weight < 50 or >100) and date_naissance > '01/01/2005'd;`
- B `Where sex='F' and (weight in(50, 100) and date_naissance > '01JAN2005'd;`
- C `Where sex^='M' and (weight <50 or weight >100) and date_naissance >'01JAN2005'd;`

SELECTION OBSERVATIONS

REPONSE

Sachant que:

Sex est variable caractere, Weight variable numerique, date_naissance variable numerique

Je veux les étudiantes filles avec un poids inferieur à 50 ou superieur à 100 et nées après le 01/01/2005

Age comprise entre 11 et 14

Choisir la bonne proposition de l'instruction WHERE :

C Where sex^='M' and(weight <50 or weight >100) and date_naissance >'01JAN2005'd;

pas M donc F entre " car constante caractère

relatif au poids hors ranges

constante de date

SELECTION OBSERVATIONS

OPERATEURS RESERVES WHERE

Les operateurs spéciaux du WHERE sont:

OPERATEUR RESERVES WHERE	DEFINITION
CONTAINS/ ?	Dans une chaine de caractère trouver une sou-chaîne, peut importe la position
LIKE	Dans une chaine de caractère on cherche une masque (avec % et _)
BETWEEN-AND	Comprise dans une intervalle (numérique ou caractère)
WHERE SAME AND/ WHERE ALSO	Permet de écrire des autres instructions WHERE dans la même étape
IS NULL / IS MISSING	Egalité à missing (numérique ou caractère)

NB: les operateurs de la liste ne pourront pas être utilisés dans des autres instructions de comparaison (IS, WHERE, UNTIL, ...)

SELECTION OBSERVATIONS

OPERATEUR RESERVE WHERE: CONTAINS

L'opérateur **CONTAINS** cherche dans toute la chaîne de caractère une sous-chaîne de caractère

La position de la sous-chaine (celle entre ' ') à l'intérieur de la valeur de la variable n'est pas importante

Il est sensible à la casse

L'abreviation mnémonique de CONTAINS est ?

```
data class;  
set sashelp.class;  
where name CONTAINS 'A';  
run;
```

	Name
1	Alfred
2	Alice

```
data class;  
set sashelp.class;  
where name CONTAINS 'a';  
run;
```

	Name
1	Barbara
2	Carol
3	James
4	Jane
5	Janet
6	Mary
7	Ronald
8	Thomas
9	William

SELECTION OBSERVATIONS

OPERATEUR RESERVE WHERE: BETWEEN AND

L'opérateur **BETWEEN AND** sélectionne les observations dont la valeur de la variable fait partie d'une gamme inclusive de valeurs

Il peut avoir des intervalles numériques et caractères (sensible à la casse)

Les extrêmes sont compris dans l'intervalle sélectionnée

```
data class;  
set sashelp.class;  
where weight BETWEEN 90 and 100;  
run;
```

Weight	
1	98
2	99.5
3	90

```
data class;  
set sashelp.class;  
where name BETWEEN 'Ja' and 'Juz';  
run;
```

Name	
1	James
2	Jane
3	Janet
4	Jeffrey
5	John
6	Joyce
7	Judy

SELECTION OBSERVATIONS

OPERATEUR RESERVE WHERE: WHERE ALSO/WHERE SAME AND

Dans un étape , data ou proc, on a le droit à écrire une seule instruction **WHERE**.

Si on a plusieurs instructions **WHERE** seulement la dernière clause **WHERE** sera prise en compte

```
data class;  
set sashelp.class;  
where sex='F';  
where age between 13 and 15;  
where name contains 'A';  
run;
```

```
74      data class;  
75      set sashelp.class;  
76      where sex='F';  
77      where age between 13 and 15;  
NOTE: La clause WHERE a été remplacée.  
78      where name contains 'A';  
NOTE: La clause WHERE a été remplacée.  
79      run;
```

NOTE: There were 2 observations read from the data set SASHELP.CLASS.
WHERE name contains 'A';

	Name	Sex
1	Alfred	M
2	Alice	F

Pour ajouter plusieurs instructions on peut utiliser **WHERE ALSO** ou **WHERE SAME AND**

```
data class;  
set sashelp.class;  
where sex='F';  
where also age between 13 and 15;  
where same and name contains 'A';  
run;
```

```
73      data class;  
74      set sashelp.class;  
75      where sex='F';  
76      where also age between 13 and 15;  
NOTE: La clause WHERE a été augmentée.  
77      where same and name contains 'A';  
NOTE: La clause WHERE a été augmentée.  
78      run;
```

NOTE: There were 1 observations read from the data set SASHELP.CLASS.
WHERE (sex='F') and (age>=13 and age<=15) and name contains 'A';

	Name	Sex	Age
1	Alice	F	13

SELECTION OBSERVATIONS

OPERATEUR RESERVE WHERE: IS MISSING/IS NULL

L'operateur **IS NULL** ou **IS MISSING** équivalents et permettent de sélectionner les observations missing.

Les négations sont : **IS NOT NULL** et **IS NOT MISSING** (pas valeurs missing)

Espressione: WHERE var_num IS NULL équivalent WHERE var_num=.

Espressione: WHERE var_car IS NULL équivalent WHERE var_car =“.”;

```
data class;  
set sashelp.class;  
where name is missing;  
run;
```

```
73      data class;  
74      set sashelp.class;  
75      where name is missing;  
76      run;
```

NOTE: There were 0 observations read from the data set SASHELP.CLASS.
WHERE name is null;
NOTE: The data set WORK.CLASS has 0 observations and 5 variables.

```
proc print data=sashelp.class;  
where name is not null;  
run;
```

Obs.	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

SELECTION OBSERVATIONS

OPERATEUR RESERVE WHERE: LIKE

L'opérateur LIKE cherche dans la chaîne de caractère une sous-chaine en rapport à une masque (pattern) spécifique.

Deux caractères spéciaux permettent de déclarer le pattern:

- % spécifique n'importe quel nombre de caractère peut occuper cette position
- _ spécifique qu'il y a exactement un caractère qui doit occuper cette position

```
data class;  
set sashelp.class;  
where name like '_o%e';  
run;
```

*commente par n'importe quelle lettre
En deuxième position la lettre O
Peut importe quelle lettre par la suite
Le nom doit terminer par e*

	Name
1	Joyce
2	Louise

SELECTION OBSERVATIONS

INSTRUCTION WHERE: QUESTION

Sachant que:

La structure de la variable NOM est Prenom, Nom

Je veux garder que les observations ou le prénom commence avec la lettre M

Choisir la bonne proposition de l'instruction WHERE :

A `Where name LIKE '_, M_';`

B `Where name like '%, M%'`

C `Where name LIKE '_, M_';`

D `Where name LIKE '%, M_';`



NOM

Stephano, Sandrina

Martinez, Cynthia

Sepke, Markus

Mccluney, Michael

SELECTION OBSERVATIONS

INSTRUCTION WHERE: REPONSE

Sachant que:

La structure de la variable NOM est Prénom, Nom

Je veux garder que les observations ou le prénom commence avec la lettre M

Choisir la bonne proposition de l'instruction WHERE :

B

```
Where name like '%, M%'
```

*commente par n'importe quelle et combien des lettres
Il trouve une , et un espace
Cherche M
Et n'importe quelles lettres et quel nombre des lettres*

NOM

Stephano, Sandrina

Martinez, Cynthia

Sepke, Markus

Mccluney, Michael

COMPILATION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
where sex eq 'F' and age between 14 and 16;  
run;
```

SAS crée un espace dans la **WORK**
pour y stocker la future table
CLASS

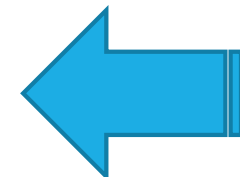
COMPILATION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
where sex eq 'F' and age between 14 and 16;  
run;
```

SAS rencontre des variables via l'instruction SET
SAS récupère le bloque descripteur des variables de
la table **SASHELP.CLASS**
c'est la première fois que il va lire ces variables
pendant la compilation de cette étape data et donc il
va les créer dans le PDV
SAS va copier toutes les variables avec leur
attributs dans l'ordre de la table source.

Program Data Vector				
NAME	SEX	AGE	HEIGHT	WEIGHT
\$ 8	\$ 1	N 8	N 8	N 8



Bloque descripteur des variables de SASHELP.CLASS

Liste alphabétique des variables et des attributs

#	Variable	Type	Long.
3	Age	Num.	8
4	Height	Num.	8
1	Name	Texte	8
2	Sex	Texte	1
5	Weight	Num.	8

COMPILATION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
where sex eq 'F' and age between 14 and 16;  
run;
```

WHERE est une instruction de exécution car elle travail sur les données donc elle est ignoré en phase de compilation

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT
\$ 8	\$ 1	N 8	N 8	N 8

COMPILATION

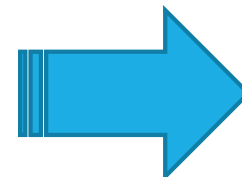
EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
where sex eq 'F' and age between 14 and 16;  
run;
```

RUN indique la fin de l'étape et donc la fin de la compilation

SAS va copier le PDV dans l'espace **WORK.CLASS**, sera son bloque descripteur des variables
SAS peut commencer l'exécution

Program Data Vector				
NAME	SEX	AGE	HEIGHT	WEIGHT
\$ 8	\$ 1	N 8	N 8	N 8



Blocue descripteur des variables de WORK.CLASS		
NOM	TYPE	LONG
NAME	\$	8
SEX	\$	1
AGE	N	8
HEIGHT	N	8
WEIGHT	N	8

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
where sex eq 'F' and age between 14 and 16;  
run;
```

SAS crée les variables `_N_` et `_ERROR_`
SAS initialise toutes les variables du PDV à
missing

Program Data Vector						
NAME \$ 8	SEX \$ 1	AGE N 8	HEIGHT N 8	WEIGHT N 8	_N_ N 8	_ERROR_ N 1
		.	.	.	1	0

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
where sex eq 'F' and age between 14 and 16;  
run;
```

SET regarde dans
SASHELP.CLASS la
ligne que vérifie la
condition **WHERE**

Zone des données de
SASHELP.CLASS

Obs.	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	N 8	N 1
Carol	F	14	62.8	102.5	1	0

NB: l'instruction **WHERE** est une filtre en entrées car l'exécution **_N_=1** est faite seulement quand
Les conditions du **WHERE** sont vérifiées

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

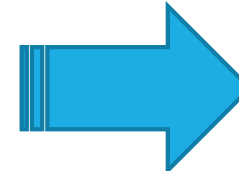


```
data class;  
set sashelp.class;  
where sex eq 'F' and age between 14 and 16;  
run;
```

Fin exécution `_N_=1`

Sortie implicite, le PDV écrit la première observation dans la table **WORK.CLASS** (Les variables `_N_` et `_ERROR_` ne sont pas copiées)
Retour implicite, on recommence une autre boucle, donc `_N_=2`

Program Data Vector						
NAME \$ 8	SEX \$ 1	AGE N 8	HEIGHT N 8	WEIGHT N 8	_N_ N 8	_ERROR_ N 1
Carol	F	14	62.8	102.5	1	0



Zone de données de WORK.CLASS				
Carol	F	14	62.8	102.5

ETAPE DATA RESULTAT

```
73      data class;  
74      set sashelp.class;  
75      where sex eq 'F' and age between 14 and 16;  
76      run;
```

NOTE: There were 4 observations read from the data set SASHELP.CLASS.
WHERE (sex='F') and (age>=14 and age<=16);

NOTE: The data set WORK.CLASS has 4 observations and 5 variables.

NOTE: DATA statement used (Total process time):

real time 0.02 seconds

cpu time 0.00 seconds

Fenêtre
Journal

CODEJOURNALRESULTATSDONNEES EN SORTIE

Table : WORK.CLASSAfficher : Libellés de la colonne

Colonne(s) : Sélectionner tout

Lignes totales : 4Colonne(s) totales : 5

Lignes 1-4

	Name	Sex	Age	Height	Weight
1	Carol	F	14	62.8	100.5
2	Janet	F	15	62.5	110.0
3	Judy	F	14	64.3	90.0
4	Mary	F	15	66.5	108.0

Propriété	Valeur
Libellé	Weight
Nom	Weight
Longueur	8
Type	Numérique
Format	

Fenêtre
Données en
Sortie

Si on veut contrôler le bloque
descripteur de notre nouvelle table:
`proc contents data=class;`
`run;`

Si on veut afficher un rapport de notre
nouvelle table:
`proc print data=class;`
`run;`

LIRE ET CREER UNE TABLE SAS

CREATION VARIABLE

```
DATA SAS-data-set-output;  
SET SAS-data-set-input;  
Nom_variable=expression;  
RUN;
```

Un simple `=` permet de créer et affecter un valeur à une nouvelle variable dans une étape data

L'instruction est composée:

- d'un nom de la variable SAS que devra suivre les règles syntaxiques standard des noms dans SAS

Et d'une expression:

- Constante caractère, numérique, date, heure, date&heure
- Variable numérique, caractère, un calcul
- Une fonctions SAS

En phase de compilation la variable est crée dans le PDV la première fois que elle est lu dans le programme

En phase de exécution les nouvelles variables vont être initialisées à missing à chaque nouvelle interaction de l'exécution (`_N_`)

LIRE ET CREER UNE TABLE SAS

CREATION VARIABLE

```
DATA SAS-data-set-output;  
SET SAS-data-set-input;  
Nom_variable=expression;  
RUN;
```

Une variable SAS est composée des attributs obligatoires et facultatifs.

Il faut déclarer les noms des nouvelles variables

Le type et la longueur peuvent être déclarés aussi (instruction LENGTH) sinon seront déduits par l'instruction d'assignation

RAPPEL: Les noms SAS (variables, tables,...) doivent suivre des règles syntaxiques:

- Ils peuvent compter de 1 à 32 caractères
- Doivent commencer par une lettre ou un _ (souligné du 8)
- Les caractères suivantes (à partir du deuxième) peuvent être des lettres, des chiffres ou des _
- Vous pouvez mélanger la casse des lettres (majuscules et minuscules)
- SAS n'est pas sensible à la casse

RAPPEL: Les attributs d'une variable SAS:

- Obligatoires: nom, type, longueur
- Facultatifs: format, label

LIRE ET CREER UNE TABLE SAS

OPERATEURS

Dans une variable SAS on peut faire les calculs avec les symboles suivantes:

SYMBOLE	DEFINITION	PRIORITE
**	Puissance	I
*	Multiplication	II
/	Division	III
+	Addition	III
-	Soustraction	III

Utiliser des **parenthèses pour clarifier ou modifier l'ordre des opérations** dans une expression arithmétique, les parenthèses plus imbriquées seront résolues en premières.

Pour toute expression arithmétique incluant une valeur manquante, le résultat sera une valeur manquante, mais ils existent des fonctions qui pourront aussi gérer les valeurs missing.

LIRE ET CREER UNE TABLE SAS

OPERATEURS ET MISSING

Une valeur . (missing) doit être considérée comme un $-\infty$ et donc n'importe quel opérateur nous pouvons utiliser donnera toujours un résultat missing.

Pour ignorer les valeurs missing il faudra utiliser les fonctions

X	Y	Z
N 8	N 8	N 8
.	4	10

Obs.	x	y	z	Somme_oper	Somme_fonc	Moyenne_oper	Moyenne_fonc
1	.	4	10	.	14	.	7

```
data test;  
set test;  
Somme_oper=(x+y+z) ;  
Somme_fonc=sum(x,y,z) ;  
Moyenne_oper=(x+y+z)/3 ;  
Moyenne_fonc=mean(x,y,z) ;  
run ;
```

SELECTION DES OBSERVATIONS

CREATION VARIABLES SAS

```
data class;  
set sashelp.class;  
Cycle='College';  
NB_eleves =19;  
Date_rentree="03SEP2018"d;  
Heure_rentree="10:00:00"t;  
Nbj_rentree=intck("day", date_rentree, today());  
Prenom=name;  
run;
```

Constante caractère

Constante numérique

Constante date

Constante time

Résultat fonctions **INTCK**

variable

NB: La fonction TODAY() récupère la date système en nombre de jours à partir du 01/01/1960

NB: la fonction INTCK() permet de calculer la difference entre deux dates dans une unité demandée (ici 'day')

ETAPE DATA RESULTAT

```
74      data class;  
75      set sashelp.class;  
76      Cycle='College';  
77      NB_eleves=19;  
78      Date_rentree="03SEP2018"d;  
79      Heure_rentree="10:00:00"t;  
80      Nbj_rentree= intck("day", date_rentree, today());  
81      Prenom=name;  
82      run;
```

NOTE: There were 19 observations read from the data set SASHELP.CLASS.

NOTE: The data set WORK.CLASS has 19 observations and 11 variables.

NOTE: DATA statement used (Total process time):

real time 0.00 seconds
cpu time 0.00 seconds

Fenêtre
Journal

CODE		JOURNAL	RESULTATS	DONNEES EN SORTIE	
Table :		WORK.CLASS	Afficher :	Libellés de la colonne	Filter : (néant)
Colonnes		Lignes totales : 19 Colonnes totales : 11			
<input checked="" type="checkbox"/> Sélectionner tout		Lignes 1-19			
<input checked="" type="checkbox"/> Height					
<input checked="" type="checkbox"/> Weight					
<input checked="" type="checkbox"/> Cycle					
<input checked="" type="checkbox"/> NB_eleves					
<input checked="" type="checkbox"/> Date_rentree					
<input checked="" type="checkbox"/> Heure_rentree					
<input checked="" type="checkbox"/> Nbj_rentree					
<input checked="" type="checkbox"/> Prenom					
Propriété	Valeur				
Libellé	Name				
Nom	Name				
Longueur	8				
Type	Alphanum				
Format					
		NB_eleves	Date_rentree	Heure_rentree	Nbj_rentree Prenom
		19	21430	36000	91 Alfred
		19	21430	36000	91 Alice
		19	21430	36000	91 Barbara
		19	21430	36000	91 Carol
		19	21430	36000	91 Henry
		19	21430	36000	91 James
		19	21430	36000	91 Jane
		19	21430	36000	91 Janet
		19	21430	36000	91 Jeffrey
		19	21430	36000	91 John
		19	21430	36000	91 Joyce
		19	21430	36000	91 Judy
		19	21430	36000	91 Louise
		19	21430	36000	91 Mary
		19	21430	36000	91 Philip

Fenêtre
Données en
Sortie

COMPILATION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Cycle='College';  
NB_eleves =19;  
Date_rentree="03SEP2018"d;  
Heure_rentree="10:00:00"t;  
Nbj_rentree=intck("day", date_rentree, today());  
Prenom=name;  
run;
```

SAS crée un espace dans la
WORK pour y stocker la future
table **CLASS**

COMPILATION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Cycle='College';  
NB_eleves =19;  
Date_rentree="03SEP2018"d;  
Heure_rentree="10:00:00"t;  
Nbj_rentree= intck("day", date_rentree, today());  
Prenom=name;  
run;
```

SAS récupère le bloque descripteur des variables de la table **SASHELP.CLASS** c'est la première fois que il va lire ces variables et donc il va créer le PDV et il va copier l'ordre, les attributs obligatoires et facultatifs

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT
\$ 8	\$ 1	N 8	N 8	N 8

COMPILATION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Cycle='College';  
NB_eleves =19;  
Date_rentree="03SEP2018"d;  
Heure_rentree="10:00:00"t;  
Nbj_rentree= intck("day", date_rentree, today());  
Prenom=name;  
run;
```

CYCLE variable que n'existe pas encore dans le PDV et donc elle va la créée.
Elle a un valeur entre '' donc sera une variable caractère \$
La longueur de cette variable est la longueur de l'initialisation donc College=7

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	CYCLE
\$ 8	\$ 1	N 8	N 8	N 8	\$ 7

COMPILATION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Cycle='College';  
NB_eleves=19;  
Date_rentree="03SEP2018"d;  
Heure_rentree="10:00:00"t;  
Nbj_rentree= intck("day", date_rentree, today());  
Prenom=name;  
run;
```

NB_ELEVES variable que n'existe pas encore dans le PDV et donc elle va la créer. Elle a une valeur sans '' donc sera une variable numérique **N**. La longueur d'une variable numérique est toujours de **8**.

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	CYCLE	NB_ELEVES
\$ 8	\$ 1	N 8	N 8	N 8	\$ 7	N 8

COMPILATION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Cycle='College';  
NB_eleves=19;  
Date_rentree="03SEP2018"d;  
Heure_rentree="10:00:00"t;  
Nbj_rentree= intck("day", date_rentree, today());  
Prenom=name;  
run;
```

DATE_RENTREE variable que n'existe pas encore dans le PDV et donc elle va la créée.
Elle a un valeur avec ' 'D' que indique une constante de date
Elle sera interprétée comme une variable numérique **N**
La longueur d'une variable numérique est toujours de **8**

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	CYCLE	NB_ELEVES	DATE_RENTREE
\$ 8	\$ 1	N 8	N 8	N 8	\$ 7	N 8	N 8

COMPILATION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Cycle='College';  
NB_eleves=19;  
Date_rentree="03SEP2018"d;  
Heure_rentree="10:00:00"t;  
Nbj_rentree= intck("day", date_rentree, today());  
Prenom=name;  
run;
```

HEURE_RENTREE variable que n'existe pas encore dans le PDV et donc elle va la créer.
Elle a une valeur avec ' 'T' que indique une constante time
Elle sera interprétée comme une variable numérique **N**
La longueur d'une variable numérique est toujours de **8**

Program Data Vector								
NAME	SEX	AGE	HEIGHT	WEIGHT	CYCLE	NB_ELEVES	DATE_RENTREE	HEURE_RENTREE
\$ 8	\$ 1	N 8	N 8	N 8	\$ 7	N 8	N 8	N 8

NB: la syntaxe 'heure: minute: secondes' t déclare une constante de date, un nombre de seconde à partir de minuit

COMPILATION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Cycle='College';  
NB_eleves=19;  
Date_rentree="03SEP2018"d;  
Heure_rentree="10:00:00"t;  
Nbj_rentree= intck("day", date_rentree, today());  
Prenom=name;  
run;
```

NBJ_RENTREE variable que n'existe pas encore dans le PDV et donc elle va la créer. Elle est le résultat d'une fonction **INTCK** donc numérique **N**
La longueur d'une variable numérique est toujours de **8**

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	CYCLE	NB_ELEVES	DATE_RENTREE	HEURE_RENTREE	NBJ_RENTREE
\$ 8	\$ 1	N 8	N 8	N 8	\$ 7	N 8	N 8	N 8	N 8

NB: La fonction TODAY() récupéré la date système en nombre de jours à partir du 01/01/1960

NB: la fonction INTCK() permet de calculer la différence entre deux dates dans une unité demandée (ici 'day')

COMPILATION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Cycle='College';  
NB_eleves=19;  
Date_rentree="03SEP2018"d;  
Heure_rentree="10:00:00"t;  
Nbj_rentree= intck("day", date_rentree, today());  
Prenom=name;  
run;
```

PRENOM variable que n'existe pas encore dans le PDV et donc elle va la créer.

Elle est la copie de la variable **NAME** donc elle va hériter de ces attributs donc \$8

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	CYCLE	NB_ELEVES	DATE_RENTREE	HEURE_RENTREE	NBJ_RENTREE	PRENOM
\$ 8	\$ 1	N 8	N 8	N 8	\$ 7	N 8	N 8	N 8	N 8	\$ 8

COMPILATION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Cycle='College';  
NB_eleves=19;  
Date_rentree="03SEP2018"d;  
Heure_rentree="10:00:00"t;  
Nbj_rentree= intck("day", date_rentree, today());  
Prenom=name;  
run;
```

SAS termine la compilation
Il va copier le PDV dans l'espace pour **WORK.CLASS**
SAS peut commencer l'exécution

Bloque descripteur des variables de **WORK.CLASS**

NOM	TYPE	LONG
NAME	\$	8
SEX	\$	1
AGE	N	8
HEIGHT	N	8
WEIGHT	N	8
CYCLE	\$	7
NB_ELEVES	N	8
DATE_RENTREE	N	8
HEURE_RENTREE	N	8
NBJ_RENTREE	N	8
PRENOM	\$	8



Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	CYCLE	NB_ELEVES	DATE_RENTREE	HEURE_RENTREE	NBJ_RENTREE	PRENOM
\$ 8	\$ 1	N 8	N 8	N 8	\$ 7	N 8	N 8	N 8	N 8	\$ 8

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Cycle='College';  
NB_eleves=19;  
Date_rentree="03SEP2018"d;  
Heure_rentree="10:00:00"t;  
Nbj_rentree= intck("day", date_rentree, today());  
Prenom=name;  
run;
```

SAS crée les variables `_N_` et `_ERROR_`
SAS initialise toutes les variables du PDV à missing

Program Data Vector

NAME \$ 8	SEX \$ 1	AGE N 8	HEIGHT N 8	WEIGHT N 8	CYCLE \$ 7	NB_ELEVES N 8	DATE_RENTREE N 8	HEURE_RENTREE N 8	NBJ_RENTREE N 8	PRENOM \$ 8	_N_ N 8	_ERROR_ N 1
			1	0

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Cycle='College';  
NB_eleves=19;  
Date_rentree="03SEP2018"d;  
Heure_rentree="10:00:00"t;  
Nbj_rentree= intck("day", date_rentree, today());  
Prenom=name;  
run;
```

SET permet de copier une ligne de la table en input **SASHELP.CLASS** dans le PDV

Zone des données de SASHELP.CLASS

Obs.	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

Program Data Vector

NAME \$ 8	SEX \$ 1	AGE N 8	HEIGHT N 8	WEIGHT N 8	CYCLE \$ 7	NB_ELEVES N 8	DATE_RENTREE N 8	HEURE_RENTREE N 8	NBJ_RENTREE N 8	PRENOM \$ 8	_N_ N 8	_ERROR_ N 1
Alfred	M	14	69.0	112.5			1	0

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Cycle='College';  
NB_eleves=19;  
Date_rentree="03SEP2018"d;  
Heure_rentree="10:00:00"t;  
Nbj_rentree= intck("day", date_rentree, today());  
Prenom=name;  
run;
```

Affectation de la valeur **College**
à la variable **CYCLE**

Program Data Vector

NAME \$ 8	SEX \$ 1	AGE N 8	HEIGHT N 8	WEIGHT N 8	CYCLE \$ 7	NB_ELEVES N 8	DATE_RENTREE N 8	HEURE_RENTREE N 8	NBJ_RENTREE N 8	PRENOM \$ 8	_N_ N 8	_ERROR_ N 1
Alfred	M	14	69.0	112.5	College		1	0

EXÉCUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Cycle='College';  
NB_eleves=19;  
Date_rentree="03SEP2018"d;  
Heure_rentree="10:00:00"t;  
Nbj_rentree= intck("day", date_rentree, today());  
Prenom=name;  
run;
```

Affectation de la valeur **19** à la variable **NB_ELEVES**

Program Data Vector

NAME \$ 8	SEX \$ 1	AGE N 8	HEIGHT N 8	WEIGHT N 8	CYCLE \$ 7	NB_ELEVES N 8	DATE_RENTREE N 8	HEURE_RENTREE N 8	NBJ_RENTREE N 8	PRENOM \$ 8	_N_ N 8	_ERROR_ N 1
Alfred	M	14	69.0	112.5	College	19	.	.	.		1	0

EXÉCUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Cycle='College';  
NB_eleves=19;  
Date_rentree="03SEP2018"d;  
Heure_rentree="10:00:00"t;  
Nbj_rentree= intck("day", date_rentree, today());  
Prenom=name;  
run;
```

Calcule du nombre des jours entre la date de référence **01/01/1960** et la constante date **03/09/2018** et stockage dans la variable **DATE_RENTREE**

Program Data Vector

NAME \$ 8	SEX \$ 1	AGE N 8	HEIGHT N 8	WEIGHT N 8	CYCLE \$ 7	NB_ELEVES N 8	DATE_RENTREE N 8	HEURE_RENTREE N 8	NBJ_RENTREE N 8	PRENOM \$ 8	_N_ N 8	_ERROR_ N 1
Alfred	M	14	69.0	112.5	College	19	21430	.	.		1	0

EXÉCUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Cycle='College';  
NB_eleves=19;  
Date_rentree="03SEP2018"d;  
Heure_rentree="10:00:00"t;  
Nbj_rentree= intck("day", date_rentree, today());  
Prenom=name;  
run;
```

Calcule du nombre des secondes
entre **00:00** et **10:00:00** et va à
l'affecter à la constante
HEURE_RENTREE donc 36000
secondes

Program Data Vector

NAME \$ 8	SEX \$ 1	AGE N 8	HEIGHT N 8	WEIGHT N 8	CYCLE \$ 7	NB_ELEVES N 8	DATE_RENTREE N 8	HEURE_RENTREE N 8	NBJ_RENTREE N 8	PRENOM \$ 8	_N_ N 8	_ERROR_ N 1
Alfred	M	14	69.0	112.5	College	19	21430	36000	.		1	0

EXÉCUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Cycle='College';  
NB_eleves=19;  
Date_rentree="03SEP2018"d;  
Heure_rentree="10:00:00"t;  
Nbj_rentree= intck("day", date_rentree, today());  
Prenom=name;  
run;
```

La fonction plus imbriquée **TODAY()** va être résolue d'abord avec nb des jours entre **01/01/1960** et **03/12/2018** (**21521** jours)
En suite la fonction **INTCK** pour calculer l'intervalle, en jours '**day**' entre la **DATE_RENTREE** et résultat sera **91**

Program Data Vector

NAME \$ 8	SEX \$ 1	AGE N 8	HEIGHT N 8	WEIGHT N 8	CYCLE \$ 7	NB_ELEVES N 8	DATE_RENTREE N 8	HEURE_RENTREE N 8	NBJ_RENTREE N 8	PRENOM \$ 8	_N_ N 8	_ERROR_ N 1
Alfred	M	14	69.0	112.5	College	19	21430	36000	91		1	0

EXÉCUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Cycle='College';  
NB_eleves=19;  
Date_rentree="03SEP2018"d;  
Heure_rentree="10:00:00"t;  
Nbj_rentree= intck("day", date_rentree, today());  
Prenom=name;  
run;
```


SAS récupère du PDV la valeur de la variable **NAME** et va le copier dans la variable **PRENOM**

Program Data Vector

NAME \$ 8	SEX \$ 1	AGE N 8	HEIGHT N 8	WEIGHT N 8	CYCLE \$ 7	NB_ELEVES N 8	DATE_RENTREE N 8	HEURE_RENTREE N 8	NBJ_RENTREE N 8	PRENOM \$ 8	_N_ N 8	_ERROR_ N 1
Alfred	M	14	69.0	112.5	College	19	21430	36000	91	Alfred	1	0

EXÉCUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION



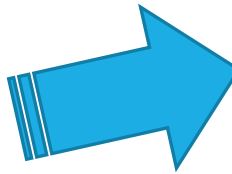
```
data class;  
set sashelp.class;  
Cycle='College';  
NB_eleves=19;  
Date_rentree="03SEP2018"d;  
Heure_rentree="10:00:00"t;  
Nbj_rentree= intck("day", date_rentree, today());  
Prenom=name;  
run;
```

Fin exécution `_N_=1`
Sortie implicite, le PDV écrit
l'observation 1 dans la table
WORK.CLASS
Retour implicite, on recommence
une autre boucle, donc `_N_=2`

Zone de données de WORK.CLASS

Alfred	M	14	69.0	112.5	College	19	21430	36000	91	Alfred
--------	---	----	------	-------	---------	----	-------	-------	----	--------

Program Data Vector



NAME \$ 8	SEX \$ 1	AGE N 8	HEIGHT N 8	WEIGHT N 8	CYCLE \$ 7	NB_ELEVES N 8	DATE_RENTREE N 8	HEURE_RENTREE N 8	NBJ_RENTREE N 8	PRENOM \$ 8	_N_ N 8	_ERROR_ N 1
Alfred	M	14	69.0	112.5	College	19	21430	36000	91	Alfred	1	0

EXÉCUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Cycle='College';  
NB_eleves=19;  
Date_rentree="03SEP2018"d;  
Heure_rentree="10:00:00"t;  
Nbj_rentree= intck("day", date_rentree, today());  
Prenom=name;  
run;
```

*_N_=2 et _ERROR_ remise 0
SAS garde dans le PDV la valeur de
anciennes variables (variables
héritées du **SET**)
SAS initialise toutes les nouvelles
variables (créé au cours de l'étape
data) du PDV à missing*

Program Data Vector

NAME \$ 8	SEX \$ 1	AGE N 8	HEIGHT N 8	WEIGHT N 8	CYCLE \$ 7	NB_ELEVES N 8	DATE_RENTREE N 8	HEURE_RENTREE N 8	NBJ_RENTREE N 8	PRENOM \$ 8	_N_ N 8	_ERROR_ N 1
Alfred	M	14	69.0	112.5			2	0

EXÉCUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Cycle='College';  
NB_eleves=19;  
Date_rentree="03SEP2018"d;  
Heure_rentree="10:00:00"t;  
Nbj_rentree= intck("day", date_rentree, today());  
Prenom=name;  
run;
```

SET permet de copier une ligne de la table en input **SASHELP.CLASS** dans le PDV

Zone des données de SASHELP.CLASS

Obs.	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

Program Data Vector

NAME \$ 8	SEX \$ 1	AGE N 8	HEIGHT N 8	WEIGHT N 8	CYCLE \$ 7	NB_ELEVES N 8	DATE_RENTREE N 8	HEURE_RENTREE N 8	NBJ_RENTREE N 8	PRENOM \$ 8	_N_ N 8	_ERROR_ N 1
Alice	F	13	56.5	84.0			2	0

ETAPE DATA RESULTAT

```
74      data class;  
75      set sashelp.class;  
76      Cycle='College';  
77      NB_eleves=19;  
78      Date_rentree="03SEP2018"d;  
79      Heure_rentree="10:00:00"t;  
80      Nbj_rentree= intck("day", date_rentree, today());  
81      Prenom=name;  
82      run;
```

NOTE: There were 19 observations read from the data set SASHELP.CLASS.

NOTE: The data set WORK.CLASS has 19 observations and 11 variables.

NOTE: DATA statement used (Total process time):

real time 0.00 seconds
cpu time 0.00 seconds

Fenêtre
Journal

CODE		JOURNAL	RESULTATS	DONNEES EN SORTIE	
Table :		WORK.CLASS	Afficher :	Libellés de la colonne	Filter : (néant)
Colonnes		Lignes totales : 19 Colonnes totales : 11			
<input checked="" type="checkbox"/> Sélectionner tout		Lignes 1-19			
<input checked="" type="checkbox"/> Height					
<input checked="" type="checkbox"/> Weight					
<input checked="" type="checkbox"/> Cycle					
<input checked="" type="checkbox"/> NB_eleves					
<input checked="" type="checkbox"/> Date_rentree					
<input checked="" type="checkbox"/> Heure_rentree					
<input checked="" type="checkbox"/> Nbj_rentree					
<input checked="" type="checkbox"/> Prenom					
Propriété	Valeur				
Libellé	Name				
Nom	Name				
Longueur	8				
Type	Alphanum				
Format					
		NB_eleves	Date_rentree	Heure_rentree	Nbj_rentree Prenom
		19	21430	36000	91 Alfred
		19	21430	36000	91 Alice
		19	21430	36000	91 Barbara
		19	21430	36000	91 Carol
		19	21430	36000	91 Henry
		19	21430	36000	91 James
		19	21430	36000	91 Jane
		19	21430	36000	91 Janet
		19	21430	36000	91 Jeffrey
		19	21430	36000	91 John
		19	21430	36000	91 Joyce
		19	21430	36000	91 Judy
		19	21430	36000	91 Louise
		19	21430	36000	91 Mary
		19	21430	36000	91 Philip

Fenêtre
Données en
Sortie

SELECTION OBSERVATIONS

INSTRUCTION IF SELECTION

```
DATA SAS-data-set-output;  
SET SAS-data-set-input;  
IF operande operateur;  
RUN;
```

L'instruction **WHERE** permet de faire **un filtre en entrée** et donc de conditionner les observations que vont être chargées dans le PDV, de conséquence **il peut marcher que avec des anciennes variables** (variables héritées de SET, MERGE, ...)

Si on **crée une variable dans l'étape data** et on veut faire la sélection des observations sur le **résultat** de cette nouvelle variable il faudra utiliser **l'instruction IF sélection**.

L'instruction IF est un filtre en sortie donc il marche sur toutes variables, nouvelles ou héritées

L'instruction IF peut être utilisée que dans les étapes data, elle n'est pas acceptée dans les étapes de proc

Dans une étape data on peut utiliser une ou plusieurs instructions IF, on peut aussi utiliser un instruction WHERE et une et plusieurs instructions IF

SELECTION OBSERVATIONS

INSTRUCTION IF SELECTION

```
DATA SAS-data-set-output;  
SET SAS-data-set-input;  
IF operande opérateur;  
RUN;
```

Dans l'instruction IF nous allons mettre des expression:

- Si l'expression est vérifiée l'observation continue à être traitée dans la boucle de l'exécution
- Si l'expression n'est pas vérifiée l'instruction traitée sera le RUN, on ne fera pas la sortie implicite mais seulement le retour implicite pour charger l'observation suivante dans le PDV

L'expression est composé d'un opérande et d'un operateur

Les operateurs exclusifs du WHERE ne pourront donc pas être utilisés (like, contains, is null, is missing, between and).

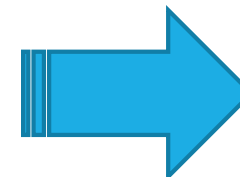
COMPILE

EXEMPLE FIN COMPILE

```
data class;  
set sashelp.class;  
Nb_majorite=(18-age);  
if sex eq 'F' and nb_majorite >=2 and nb_majorite <=4;  
Cycle='College';  
run;
```

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	NB_MAJORITE	CYCLE
\$ 8	\$ 1	N 8	N 8	N 8	N 8	\$ 7



Bloque descripteur des variables
de SASHELP.CLASS

NOM	TYPE	LONG
NAME	\$	8
SEX	\$	1
AGE	N	8
HEIGHT	N	8
WEIGHT	N	8
NB_MAJORITE	N	8
CYCLE	\$	7

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Nb_majorite=(18-age);  
if sex eq 'F' and nb_majorite >=2 and nb_majorite <=4;  
Cycle='College';  
run;
```

SAS crée les variables `_N_` et `_ERROR_`
SAS initialise toutes les variables du PDV à missing

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	NB_MAJORITE	CYCLE	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	N 8	\$ 7	N 8	N 1
			1	0

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Nb_majorite=(18-age);  
if sex eq 'F' and nb_majorite >=2 and nb_majorite <=4;  
Cycle='College';  
run;
```

Zone des données de SASHELP.CLASS

Obs.	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	NB_MAJORITE	CYCLE	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	N 8	\$ 7	N 8	N 1
Alfred	M	14	69.0	112.5	.		1	0

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION


```
data class;  
set sashelp.class;  
Nb_majorite=(18-age);  
if sex eq 'F' and nb_majorite >=2 and nb_majorite <=4;  
Cycle='College';  
run;
```

Affectation de la valeur **4** (18-14)
à la variable **NB_MAJORITE**

Program Data Vector								
NAME	SEX	AGE	HEIGHT	WEIGHT	NB_MAJORITE	CYCLE	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	N 8	\$ 7	N 8	N 1
Alfred	M	14	69.0	112.5	4		1	0

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION



```
data class;  
set sashelp.class;  
Nb_majorite=(18-age);  
if sex eq 'F' and nb_majorite >=2 and nb_majorite <=4;  
Cycle='College';  
run;
```

La condition dans **IF** n'est pas vérifiée
L'exécution **_N_=1** termine ici car on va directement au **RUN**;

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	NB_MAJORITE	CYCLE	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	N 8	\$ 7	N 8	N 1
Alfred	M	14	69.0	112.5	4		1	0

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Nb_majorite=(18-age);  
if sex eq 'F' and nb_majorite >=2 and nb_majorite <=4;  
Cycle='College';  
run;
```

*_N_=2 et _ERROR_ remise 0
SAS garde dans le PDV la valeur de
anciennes variables (variables
héritées du **SET**)
SAS initialise toutes les nouvelles
variables (créé au cours de l'étape
data) du PDV à missing*

Program Data Vector								
NAME \$ 8	SEX \$ 1	AGE N 8	HEIGHT N 8	WEIGHT N 8	NB_MAJORITE N 8	CYCLE \$ 7	_N_ N 8	_ERROR_ N 1
Alfred	M	14	69.0	112.5	.		2	0

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Nb_majorite=(18-age);  
if sex eq 'F' and nb_majorite >=2 and nb_majorite <=4;  
Cycle='College';  
run;
```

Zone des données de SASHELP.CLASS

Obs.	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	NB_MAJORITE	CYCLE	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	N 8	\$ 7	N 8	N 1
Alice	F	13	56.5	84.0	.		2	0

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Nb_majorite=(18-age);  
if sex eq 'F' and nb_majorite >=2 and nb_majorite <=4;  
Cycle='College';  
run;
```


Affectation de la valeur 5 (18-13)
à la variable **NB_MAJORITE**

Program Data Vector								
NAME	SEX	AGE	HEIGHT	WEIGHT	NB_MAJORITE	CYCLE	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	N 8	\$ 7	N 8	N 1
Alice	F	13	56.5	84.0	5		2	0

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Nb_majorite=(18-age);  
if sex eq 'F' and nb_majorite >=2 and nb_majorite <=4;  
Cycle='College';  
run;
```



La condition dans **IF** n'est pas vérifiée

L'exécution **_N_=2** termine ici car on va directement au **RUN**;

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	NB_MAJORITE	CYCLE	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	N 8	\$ 7	N 8	N 1
Alice	F	13	56.5	84.0	5		2	0

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Nb_majorite=(18-age);  
if sex eq 'F' and nb_majorite >=2 and nb_majorite <=4;  
Cycle='College';  
run;
```

*_N_=3 et _ERROR_ remise 0
SAS garde dans le PDV la valeur de
anciennes variables (variables
héritées du **SET**)
SAS initialise toutes les nouvelles
variables (créé au cours de l'étape
data) du PDV à missing*

Program Data Vector								
NAME \$ 8	SEX \$ 1	AGE N 8	HEIGHT N 8	WEIGHT N 8	NB_MAJORITE N 8	CYCLE \$ 7	_N_ N 8	_ERROR_ N 1
Alice	F	13	56.5	84.0	.		3	0

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Nb_majorite=(18-age);  
if sex eq 'F' and nb_majorite >=2 and nb_majorite <=4;  
Cycle='College';  
run;
```

Zone des données de SASHELP.CLASS

Obs.	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	NB_MAJORITE	CYCLE	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	N 8	\$ 7	N 8	N 1
Barbara	F	13	65.5	98.0	.		3	0

EXÉCUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Nb_majorite=(18-age);  
if sex eq 'F' and nb_majorite >=2 and nb_majorite <=4;  
Cycle='College';  
run;
```

Affectation de la valeur 5 (18-13)
à la variable **NB_MAJORITE**


Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	NB_MAJORITE	CYCLE	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	N 8	\$ 7	N 8	N 1
Barbara	F	13	65.5	98.0	5		3	0

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Nb_majorite=(18-age);  
if sex eq 'F' and nb_majorite >=2 and nb_majorite <=4;  
Cycle='College';  
run;
```



La condition dans **IF** n'est pas vérifiée
L'exécution **_N_=3** termine ici car on va directement au **RUN**;

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	NB_MAJORITE	CYCLE	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	N 8	\$ 7	N 8	N 1
Barbara	F	13	65.5	98.0	5		3	0

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Nb_majorite=(18-age);  
if sex eq 'F' and nb_majorite >=2 and nb_majorite <=4;  
Cycle='College';  
run;
```

*_N_=4 et _ERROR_ remise 0
SAS garde dans le PDV la valeur de
anciennes variables (variables
héritées du **SET**)
SAS initialise toutes les nouvelles
variables (créé au cours de l'étape
data) du PDV à missing*

Program Data Vector

NAME \$ 8	SEX \$ 1	AGE N 8	HEIGHT N 8	WEIGHT N 8	NB_MAJORITE N 8	CYCLE \$ 7	_N_ N 8	_ERROR_ N 1
Barbara	F	13	65.5	98.0	.		4	0

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Nb_majorite=(18-age);  
if sex eq 'F' and nb_majorite >=2 and nb_majorite <=4;  
Cycle='College';  
run;
```

Zone des données de SASHELP.CLASS

Obs.	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

Program Data Vector

NAME	SEX	AGE	HEIGHT	WEIGHT	NB_MAJORITE	CYCLE	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	N 8	\$ 7	N 8	N 1
Carol	F	14	62.8	102.5	.		4	0

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Nb_majorite=(18-age);  
if sex eq 'F' and nb_majorite >=2 and nb_majorite <=4;  
Cycle='College';  
run;
```

Affectation de la valeur **4** (18-14)
à la variable **NB_MAJORITE**

Program Data Vector								
NAME	SEX	AGE	HEIGHT	WEIGHT	NB_MAJORITE	CYCLE	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	N 8	\$ 7	N 8	N 1
Carol	F	14	62.8	102.5	4		4	0

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION

```
data class;  
set sashelp.class;  
Nb_majorite=(18-age);  
if sex eq 'F' and nb_majorite >=2 and nb_majorite <=4;  
Cycle='College';  
run;
```

La condition dans **IF** est vérifiée
L'exécution _N_=4 continue

Program Data Vector

NAME \$ 8	SEX \$ 1	AGE N 8	HEIGHT N 8	WEIGHT N 8	NB_MAJORITE N 8	CYCLE \$ 7	_N_ N 8	_ERROR_ N 1
Carol	F	14	62.8	102.5	5		4	0

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION


```
data class;  
set sashelp.class;  
Nb_majorite=(18-age);  
if sex eq 'F' and nb_majorite >=2 and nb_majorite <=4;  
Cycle='College';  
run;
```

Affectation de la valeur College à la variable **CYCLE**

Program Data Vector								
NAME	SEX	AGE	HEIGHT	WEIGHT	NB_MAJORITE	CYCLE	_N_	_ERROR_
\$ 8	\$ 1	N 8	N 8	N 8	N 8	\$ 7	N 8	N 1
Carol	F	14	62.8	102.5	5	College	4	0

EXECUTION

EXEMPLE INSTRUCTION PAR INSTRUCTION



```
data class;  
set sashelp.class;  
Nb_majorite=(18-age);  
if sex eq 'F' and nb_majorite >=2 and nb_majorite <=4;  
Cycle='College';  
run;
```

Fin exécution `_N_=4`

Sortie implicite, le PDV écrit
l'observation 1 dans la table

WORK.CLASS

Retour implicite, on recommence
une autre boucle, donc `_N_=5`

Program Data Vector							Zone de données de WORK.CLASS		
NAME	SEX	AGE	HEIGHT	WEIGHT	NB_MAJORITE	CYCLE	_N_	_ERROR_	
\$ 8	\$ 1	N 8	N 8	N 8	N 8	\$ 7	N 8	N 1	
Carol	F	14	62.8	102.5	5	College	4	0	

ETAPE DATA RESULTAT

```
72
73     data class;
74     set sashelp.class;
75     Nb_majorite=(18-age);
76     if sex eq 'F' and nb_majorite >=2 and nb_majorite <=4;
77     Cycle='College';
78     run;
```

NOTE: There were 19 observations read from the data set SASHELP.CLASS.
NOTE: The data set WORK.CLASS has 4 observations and 7 variables.
NOTE: DATA statement used (Total process time):
 real time 0.00 seconds
 cpu time 0.01 seconds

Fenêtre
Journal

Si on veut contrôler le bloque
descripteur de notre nouvelle table:
`proc contents data=class;`
`run;`

Si on veut afficher un rapport de notre
nouvelle table:
`proc print data=class;`
`run;`

CODE	JOURNAL	RESULTATS	DONNEES EN SORTIE
Table :	WORK.CLASS	Afficher :	Libellés de la colonne
Colonnes			
Lignes totales : 4 Colonnes totales : 7			
Lignes 1-4			
<input checked="" type="checkbox"/> Sélectionner tout			
<input checked="" type="checkbox"/> Name			
<input checked="" type="checkbox"/> Sex			
<input checked="" type="checkbox"/> Age			
<input type="checkbox"/> Height			
<input type="checkbox"/> Weight			
<input checked="" type="checkbox"/> Nb_majorite			
<input checked="" type="checkbox"/> Cycle			
Propriété Valeur			
Libellé	Weight		
Nom	Weight		
Longueur	8		
Type	Numérique		

Fenêtre
Données en
Sortie

EXERCICES

SEMAINE 4



**KEEP
CALM
AND
DO YOUR
HOMEWORK**

- Déclarer la bibliothèque UT1_SAS qui pointe sur votre fichier (pour moi UT1_FOAD)
`libname UT1_SAS '/folders/myfolders/UT1_FOAD' ;`
- Afficher le bloque descripteur de la table UT1_SAS.CUSTOMER_DIM
- A partir de la table UT1_SAS.CUSTOMER_DIM créer une table appelée SELECTION_CLIENT dans la WORK
- A l'aide de la fonction INTCK() et de la fonction TODAY() calculée l'âge, en année, du client au jour d'aujourd'hui et stocké le résultat dans une nouvelle variable appelée AGE_ACTUALISEE (ex: intck("day", date_rentree, today());)
- Filtrer les clients afin de garder seulement les clients (4) que vont vérifier toutes les conditions suivantes:
 - Les clients membres d'un club , variable CUSTOMER_GROUP
 - Les clients nées dans les années 70 et 80 (entre 01/01/70 et 31/12/89)
 - Les clients ayant une voyelle comme première lettre du prénom
 - Les clients ayant une voyelle comme deuxième lettre du nom
 - Les clients ayant une âge actualisée inférieur à 40
- A partir de la table WORK.SELECTION_CLIENT créer le rapport suivant via une étape PROC

Obs.	Customer_ID	Customer_FirstName	Customer_LastName	age_actualisee	Customer_BirthDate	Customer_Group
1	34	Alvan	Goheen	34	18JAN1984	Orion Club members
2	49	Annmarie	Leveille	34	16JUL1984	Orion Club Gold members
3	19873	Avinoam	Tuvia	34	14JUN1984	Orion Club Gold members
4	70210	Alex	Santinello	32	22APR1986	Orion Club members

EXERCICES

SEMAINE 4

- BONUS: contrôler le FORMAT de la variable CUSTOMER_BIRTHDATE dans le bloque descripteur de WORK.SELECTION_CLIENT
- Dans l'étape de PROC pour afficher les résultats ajouter l'instruction FORMAT avec le bon nom du format pour avoir l'affichage de la variable CUSTOMER_BIRTHDATE comme il suit:

Obs.	Customer_ID	Customer_FirstName	Customer_LastName	age_actualisee	Customer_BirthDate	Customer_Group
1	34	Alvan	Goheen	34	18 janvier 1984	Orion Club members
2	49	Annmarie	Leveille	34	16 juillet 1984	Orion Club Gold members
3	19873	Avinoam	Tuvia	34	14 juin 1984	Orion Club Gold members
4	70210	Alex	Santinello	32	22 avril 1986	Orion Club members



**KEEP
CALM
AND
DO YOUR
HOMEWORK**

- Contrôler à nouveau dans le bloque descripteur de la table WORK.SELECTION_CLIENT le FORMAT de CUSTOMER_BIRTHDATE
- Écrivez pourquoi l'affichage de la variable CUSTOMER_BIRTHDATE est différent malgré que les métadonnées sont toujours les mêmes.



REPONSES DANS LE FORUM

Proposées vos réponses dans le forum afin de pouvoir en parler avec les autres élèves

Vos réponses ne seront pas notées, votre participation au forum oui

Posez des questions dans le forum

Lien pour l'help SAS: