Gérer vos fichiers SAS avec PROC DATASETS

Club SAS de Québec – mai 2024

Jean Hardy, Services Conseils Hardy

Complexe Jules Dallaire Tour 1 2828, boulevard Laurier Bureau 700 Québec, Qc G1V 0B9
Téléphone: (418) 626-1666 Téléphone: (514) 866-0871 www.schardy.qc.ca

Plan de la présentation

1. Contexte
2. Syntaxe
3. Modifier les attributs (variables et tables)
4. Lister les fichiers et la structure des tables
5. Copier des fichiers
6. Déplacer des fichiers
7. Renommer des fichiers
8. Échanger le nom de deux fichiers

Plan de la présentation (suite)

9. Concaténer des tables
10. Dupliquer une table de la bibliothèque
11. Attributs personnalisés
12. Détruire des fichiers
13. Autres tâches
14. Exemple intégratif

    Conclusion

1. Contexte

- Souvent présentée comme une simple alternative à d'autres procédures SAS de gestion de données:

PROC APPEND	**PROC COPY**
PROC CONTENTS	**PROC DELETE**

- Permet des tâches difficiles voire impossibles à réaliser via **DATA** ou **PROC SQL** (attributs personnalisés, etc.)
- Parfois plus efficace que DATA car ne lit pas les observations
- Parfois inutilisable avec des tables d'une CASLIB, sous Viya

1. Contexte (suite)

- Bibliothèque SAS peut contenir, outre les tables SAS: vues, catalogues, index, gabarits (« templates »), etc.

- Vue SAS → table « virtuelle » qui ne contient pas de données – seulement les instructions pour les créer :

```
DATA WH.EXPENSES(DROP=N_STUD) / VIEW=WH.EXPENSES;
  SET EDUC.EXPENSES(KEEP=SCHOOL  TEACHER  CLASS
                         EXPENSES  N_STUD);
  EXP_PCAP = EXPENSES/N_STUD;
  FORMAT EXP_PCAP DOLLAR4.;
  WHERE SCHOOL NE "Dekker";
RUN;
```

1. Contexte (suite)

- Vues SAS souvent traitées comme des tables par **DATASETS**

- Index permet l'emploi d'un **BY** dans le reste des traitements, sans recourir au tri préalable et accélère certains filtrages – index traités en même temps que la table lors de:
  - ❑copie
  - ❑déplacement
  - ❑changement de nom
  - ❑destruction

2. Syntaxe

```
PROC DATASETS  LIBRARY=bibliothèque NOLIST;
    énoncé (tel que MODIFY, COPY, APPEND, CONTENTS, etc.);
    énoncé ;
    RUN;
    énoncé(s);
QUIT;
```

- **NOLIST** supprime la liste initiale des fichiers de la bibliothèque
- Plusieurs énoncés interreliés peuvent être exécutés d'un bloc, puis suivis d'un **RUN;**

3. Modifier les attributs (variables et tables)

- **MODIFY** change certains attributs de variables (nom, libellé, format, mais pas la longueur, ni le type)
- Au lieu d'utiliser:

```
DATA WH.SALES2023;
  SET WH.SALES2023;
  STATE_PROV = STATE;
  FORMAT PERIOD YYMMP7.
         ACTUAL PREDICT NLMNY12.2;
  LABEL STATE_PROV = "State / Province"
        PRODTYPE   = "Product Type";
  DROP STATE;
RUN;
```

3. Modifier les attributs (suite)

- On utilisera plutôt:

```
PROC DATASETS  LIBRARY=WH  NOLIST;
  MODIFY SALES2023;
    RENAME STATE = STATE_PROV;
    FORMAT PERIOD YYMMP7.
           ACTUAL PREDICT NLMNY12.2;
    LABEL STATE_PROV = "State / Province"
          PRODTYPE   = "Product Type";
QUIT;
```

- Beaucoup plus rapide qu'une étape DATA
- Plusieurs énoncés **MODIFY** requis si plusieurs tables altérées

3. Modifier les attributs (suite)

- **MODIFY** permet de modifier des attributs de table, dont:
  - Libellé de table (LABEL=)
  - Nombre de générations (fichiers en génération – GENNUM=)
  - Date et heure de création (rarement utilisé – DTC=)

- Exemple – fournir un libellé à la table et modifier sa date de création:

```
PROC DATASETS  LIBRARY=WH  NOLIST;
  MODIFY SALES2023(LABEL="Gross sales - using "
                         "AGG_YRLY_SALES.sas")
                 / DTC="31-DEC-2023 23:59:59"DT;
QUIT;
```

3. Modifier les attributs (suite)

➢ Contenu de la table modifiée :

Nom de la table	WH.SALES2023	Observations	11520
Type de membre	DATA	Variables	9
Moteur	V9	Index	0
Créée	2023-12-31 23:59:59	Longueur d'observation	104
Dernière modification	2024-04-13 10:23:49	Observations supprimées	0
Protection		Compressée	NON
Type de table		Triée	NON
Libellé	Gross sales - program used: AGG_YRLY_SALES.sas		
Représentation des données	WINDOWS_64		
Codage	wlatin1 Western (Windows)		

4. Lister les fichiers et la structure des tables

- Afficher la liste des fichiers d'une bibliothèque:

```
PROC DATASETS  LIBRARY=WH;
QUIT;
```

#	Nom	Type de membre	Taille du fichier	Modifié(e) le
1	CLIENTS	DATA	192KB	2024-04-14 09:59:07
	CLIENTS	INDEX	9KB	2024-04-14 09:59:07
2	DEPT	DATA	128KB	2024-04-14 09:59:07
3	EXPENSES	VIEW	9KB	2024-04-14 09:59:07
4	FORMATS	CATALOG	17KB	2024-04-14 09:59:07
5	FORMATS_OLD	CATALOG	17KB	2024-04-14 09:59:07
6	MY_TEMPLATES	ITEMSTOR	13KB	2024-04-14 09:59:07
7	SALES2019	DATA	1MB	2024-04-14 09:59:07
8	SALES2020	DATA	1MB	2024-04-14 09:59:07
9	SALES2021	DATA	1MB	2024-04-14 09:59:07

4. Lister les fichiers et la structure ...(suite)

- **CONTENTS** affiche la structure d'une ou de toutes les tables :

```
PROC DATASETS  LIBRARY=WH  NOLIST;
  CONTENTS  DATA=SALES2023;
QUIT;
```

Nom de la table	WH.SALES2023	Observations	11520
Type de membre	DATA	Variables	9
Moteur	V9	Index	0
Créée	2024-04-14 07:36:47	Longueur d'observation	104
Dernière modification	2024-04-14 07:36:47	Observations supprimées	0
Protection		Compressée	NON
Type de table		Triée	NON
Libellé			
Représentation des données	WINDOWS_64		
Codage	wlatin1 Western (Windows)		

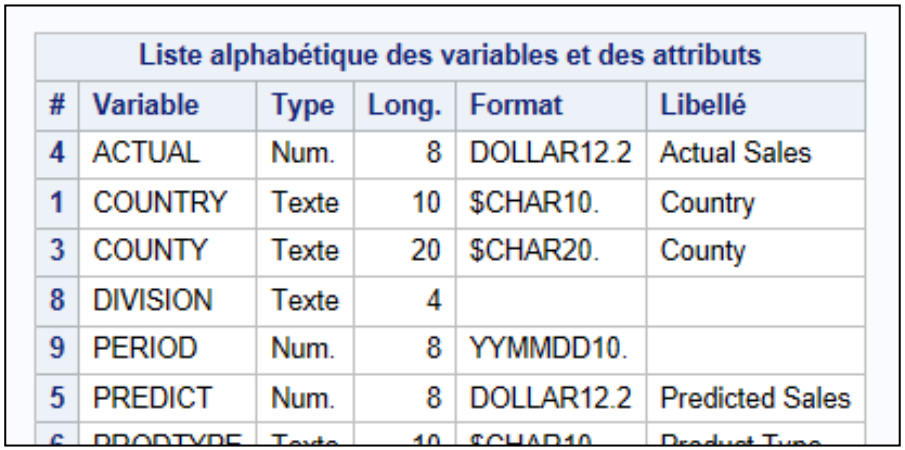
4. Lister les fichiers et la structure ...(suite)

- Suite de la structure d'une table :

Informations dépendantes de la machine/de l'hôte	
Taille de la page	65536
Nombre de pages	19
Première page de données	1
Nb max. d'obs. par page	629
Obs. sur première page de données	609
Nombre de corrections dans la table	0
ExtendObsCounter	YES
Nom du fichier	H:\DEMO-DATA\sales2023.sas7bdat
Version de création	9.0401M6
Hôte de création	X64_10PRO
[illegible]	[illegible]

4. Lister les fichiers et la structure ...(suite)

- Suite de la structure d'une table :



Liste alphabétique des variables et des attributs

#	Variable	Type	Long.	Format	Libellé
4	ACTUAL	Num.	8	DOLLAR12.2	Actual Sales
1	COUNTRY	Texte	10	$CHAR10.	Country
3	COUNTY	Texte	20	$CHAR20.	County
8	DIVISION	Texte	4		
9	PERIOD	Num.	8	YYMMDD10.	
5	PREDICT	Num.	8	DOLLAR12.2	Predicted Sales
6	PRODTYPE	Texte	10	$CHAR10.	Product Type

- Paramètre **OUT=** place les attributs dans une table, comme un dictionnaire de données

4. Lister les fichiers et la structure ...(suite)

- Lister la structure de plusieurs tables : il faut répéter l'énoncé **CONTENTS** …

```
PROC DATASETS  LIBRARY=WH  NOLIST;
  CONTENTS  DATA=SALES2023;
  CONTENTS  DATA=CLIENTS;
QUIT;
```

ou utiliser :

```
PROC DATASETS  LIBRARY=WH;
  CONTENTS  DATA=_ALL_;
QUIT;
```

5. Copier des fichiers

- **COPY** permet de copier des tables et d'autres fichiers SAS entre des bibliothèques
- Exemple - au lieu de copier des tables avec :

```
DATA WORK.CLIENTS(INDEX=(index-recréés…));
   SET WH.CLIENTS;
RUN;

DATA WORK.SALES2023;
   SET WH.SALES2023;
RUN;

DATA WORK.DEPT;
   SET WH.DEPT;
RUN;
```

5. Copier des fichiers (suite)

- Utiliser plutôt :

```
PROC DATASETS  LIBRARY=WH  NOLIST;
  COPY OUT=WORK;
    SELECT CLIENTS  SALES2023  DEPT;
QUIT;
```

```
27          PROC DATASETS  LIBRARY=WH  NOLIST;
28            COPY OUT=WORK;
29              SELECT CLIENTS  SALES2023  DEPT;
NOTE: Copying WH.CLIENTS to WORK.CLIENTS (memtype=DATA).
NOTE: Simple index CL_NAME has been defined.
NOTE: There were 14 observations read from the data set WH.CLIENTS.
NOTE: The data set WORK.CLIENTS has 14 observations and 6 variables
NOTE: Copying WH.SALES2023 to WORK.SALES2023 (memtype=DATA).
NOTE: There were 11520 observations read from the data set WH.SALES
NOTE: The data set WORK.SALES2023 has 11520 observations and 9 var
NOTE: Copying WH.DEPT to WORK.DEPT (memtype=DATA)
```

5. Copier des fichiers (suite)

- Copier entre plusieurs bibliothèques :

```
PROC DATASETS  LIBRARY=WH  NOLIST;
  COPY  OUT=WORK;
    SELECT CLIENTS  SALES2023  DEPT;
  COPY  IN=WORK  OUT=ARCHIVES;
    SELECT  liste-de-tables…;
QUIT;
```

- Copier une série de fichiers :

```
    SELECT  SALES2019-SALES2021  DEPT;
    SELECT  FORM:;
```

5. Copier des fichiers (suite)

- Copier tous les fichiers sauf certains :

```
PROC DATASETS  LIBRARY=WH  NOLIST;
  COPY OUT=WORK;
    EXCLUDE FORM:  EXPENSES  DEPT  MY_TEMPLATES;
QUIT;
```

5. Copier des fichiers (suite)

- Fichiers copiés portent <u>par défaut la date de la copie</u> comme date de création

- Pour préserver leur date de création, lors de la copie :

```
PROC DATASETS  LIBRARY=WH  NOLIST;
  COPY OUT=WORK  DATECOPY;
    SELECT CLIENTS  SALES2023  DEPT;
QUIT;
```

6. Déplacer des fichiers

- Procéder exactement comme pour la copie, mais ajouter le paramètre **MOVE** dans l'énoncé **COPY** :

```
PROC DATASETS  LIBRARY=WH  NOLIST;
  COPY OUT=WORK  MOVE;
    SELECT CLIENTS  SALES2023  DEPT;
QUIT;
```

7. Renommer des fichiers

- **CHANGE** permet de renommer des fichiers - remplace donc la méthode suivante (lourde avec des tables volumineuses):

```
DATA WH.SALES_CURRENT;
  SET WH.SALES2023;
RUN;

DATA WH.SALES_PREVIOUS;
  SET WH.SALES2022;
RUN;

PROC DELETE  DATA=WH.SALES2022  WH.SALES2023;
RUN;
```

7. Renommer des fichiers (suite)

- **CHANGE** permettra de coder:

```
PROC DATASETS  LIBRARY=WH  NOLIST;
  CHANGE SALES2023 = SALES_CURRENT;
  CHANGE SALES2022 = SALES_PREVIOUS;
QUIT;
```

- Un énoncé **CHANGE** par fichier à renommer

7. Renommer des fichiers (suite)

- Ne pas confondre avec **RENAME**, qui sert à plutôt renommer des variables d'une table

- Impossible de préfixer les fichiers par leur bibliothèque – l'opération vise uniquement la bibliothèque par défaut

8. Échanger le nom de deux fichiers

- Supposons la table *CLIENTS* créée en 2024-04 et une copie de sécurité nommée *CLIENTS_BKUP* créée en 2022-12 :

Library Name	Member Name	Date Created	Number of Logical Observations	Number of Variables
WH	CLIENTS	14APR2024:09:59:07	14	6
WH	CLIENTS_BKUP	31DEC2022:23:59:59	16	7

- Pour corriger une erreur, vous désirez échanger le nom des tables afin que la copie se nomme désormais *CLIENTS*

8. Échanger le nom de deux fichiers (suite)

```
PROC DATASETS  LIBRARY=WH  NOLIST;
  EXCHANGE  CLIENTS=CLIENTS_BKUP;
QUIT;
```

Library Name	Member Name	Date Created	Number of Logical Observations	Number of Variables
WH	CLIENTS	31DEC2022:23:59:59	16	7
WH	CLIENTS_BKUP	14APR2024:09:59:07	14	6

8. Échanger le nom de deux fichiers (suite)

- Comparer avec les étapes **DATA** requises pour obtenir le même résultat :

```
DATA WORK.CLIENTS;
   SET WH.CLIENTS;
RUN;

DATA WH.CLIENTS;
   SET WH.CLIENTS_BKUP;
RUN;

DATA WH.CLIENTS_BKUP;
   SET WORK.CLIENTS;
RUN;
```

9. Concaténer des tables

- **APPEND** ajoute, après les rangées d'une première table, les rangées d'une seconde :

```
PROC DATASETS  LIBRARY=WH  NOLIST;
  APPEND  BASE=ALL_SALES  DATA=WORK.MSAL
          FORCE;
QUIT;
```

- Peut remplacer :

```
DATA WH.ALL_SALES;
  SET WH.ALL_SALES  WORK.MSAL;
RUN;
```

9. Concaténer des tables (suite)

- Table secondaire peut être située dans autre bibliothèque
- **FORCE** est essentiel, si la structure des tables n'est pas identique (liste de variables, longueur, attributs, etc.) :
- Ne demande qu'une fraction du temps d'une étape **DATA** (parfois moins de 5%)
- Ne permet de concaténer que deux tables à la fois, alors qu'on en concatène plusieurs avec **SET** ou **PROC SQL**

10. Dupliquer une table de la bibliothèque

- Dupliquer → copier une table dans la même bibliothèque que celle en entrée mais en lui donnant un nouveau nom
- **COPY** ou **CHANGE** seuls ne peuvent y parvenir...
- **APPEND** le permet → agir comme si on concaténait 2 tables, mais fournir dans **BASE=** le nouveau nom (le duplicat)
- Possible parce que le paramètre **BASE=** peut nommer une table qui n'existe pas encore

10. Dupliquer une table dans la … (suite)

- Exemple – dupliquer la table *CLIENTS* dans la bibliothèque, sous le nouveau nom *CLIENTS_UPDT* :

```
PROC DATASETS  LIBRARY=WH  NOLIST;
  APPEND BASE=CLIENTS_UPDT  DATA=CLIENTS;
QUIT;
```

11. Attributs personnalisés

- Tables et variables comportent plusieurs attributs, tels que :

Attributs de table		Attributs de variable
Date de création	Date de modification	Type
Libellé de table	Encodage (latin, Utf8)	Longueur
Nombre d'observat.	Nombre de variables	Format/Informat
Nombre d'index	Version de SAS	Libellé

- Pratique de pouvoir définir d'autres attributs, permettant de classer les tables ou les variables, ou les caractériser : attributs personnalisés (« extended attributes » en anglais)

11. Attributs personnalisés (suite)

- Exemples d'attributs personnalisés de tables :
  - Programme ou utilisateur ayant créé ou mis à jour une table
  - Période couverte, tel que *2023-10* (≠ date de création)
  - Origine des données ou mécanisme de chargement
  - Attribut *Sujet* ou *Keyword* pour faciliter recherches thématiques

- Exemples d'attributs personnalisés de variables
  - Type détaillé (binaire, entier, date, heure, durée, monétaire, etc.)
  - Formule utilisée pour créer la variable
  - Rôle des variables (ventilateur, prédicteur, pondérateur, etc.)
  - Unité de mesure (kilos, mètres, hectares, kw/heure, etc.)

11. Attributs personnalisés (suite)

- Supposons les données suivantes:

	ORDER_NO	CL_FNAME	CL_LNAME	ORDER_DATE	ORDER_TIME	PRICE	SHIP_STATUS	SHIPPED	ELAPSED	SAMPLING_W
1	206194	Paul	Ryan	2015-10-27	14:46	$145.69	S	30OCT2015:04:28:00	61:42:00	0.7716770723
2	398763	Timothy	Schaffer	2016-01-07	9:14	$239.99	B	.	.	1.9766103065
3	1086673	Suzan	Fleming	2015-04-17	11:45	$176.00	S	26APR2015:06:50:00	211:05:00	1.1658231253

- Attributs personnalisés de la table (le nom et le contenu de ces attributs est sous la seule responsabilité de l'utilisateur):
  - ❑ rythme de rafraichissement des données (*REFRESH_CYCLE*)
  - ❑ programme ayant créé ou modifié la table (*CREATION_PGM*)
  - ❑ source des données (*DATA_SOURCE)*

11. Attributs personnalisés (suite)

```
PROC DATASETS  LIB=WH  NOLIST;
   MODIFY ORDERS;
      XATTR ADD DS
        REFRESH_CYCLE = "Weekly"
        CREATION_PGM  = "&_SASPROGRAMFILE"
        DATA_SOURCE   = "Oracle tables 'ERP19' & …";
QUIT;
```

- Attributs personnalisés des variables (ici aussi définis par l'utilisateur):
  - type détaillé (***DET_TYPE*** – avec des valeurs comme *INT*, *DATE*, *TIME*, *CURR*, etc.)
  - rôle de la variable (***ROLE*** – avec des valeurs comme *IDENT*, *CATEG*, *CONTI*, *WEIGHT*)

11. Attributs personnalisés (suite)

```
PROC DATASETS  LIB=WH  NOLIST ;
   MODIFY ORDERS;
     XATTR ADD VAR
       ORDER_NO    (DET_TYPE="INT"       ROLE="IDENT" )
       CL_FNAME    (DET_TYPE="CHAR"      ROLE="IDENT" )
       CL_LNAME    (DET_TYPE="CHAR"      ROLE="IDENT" )
       ORDER_DATE  (DET_TYPE="DATE"      ROLE="CATEG" )
       ORDER_TIME  (DET_TYPE="TIME"      ROLE="CATEG" )
       PRICE       (DET_TYPE="CURR"      ROLE="CONTI" )
       SHIP_STATUS (DET_TYPE="CHAR"      ROLE="CATEG" )
       SHIPPED     (DET_TYPE="TIMESTAMP" ROLE="CATEG" )
       ELAPSED     (DET_TYPE="DURATION"  ROLE="CONTI" )
       SAMPLING_W  (DET_TYPE="DOUBLE"    ROLE="WEIGHT");
QUIT;
```

11. Attributs personnalisés (suite)

- Attributs personnalisés gérés en même temps que la table (copie, déplacement, destruction)
- Accessibles par le dictionnaire de données **SQL**
- Visibles via l'énoncé ou la procédure **CONTENTS**

Liste alphabétique des attributs étendus de la table		
Attribut étendu	**Valeur numérique**	**Valeur caractère**
CREATION_PGM	.	'C:\Seminaires\Club SAS Québec & MONSUG\CUSQ-202405-P DATASETS\PROC DATASETS-PROGRAMME DÉMO.sas'
DATA_SOURCE	.	Oracle tables 'ERP19' & 'ERP24'
REFRESH_CYCLE	.	Weekly

Liste alphabétique des attributs étendus sur les variables			
Attribut étendu	**Variable de l'attribut**	**Valeur numérique**	**Valeur caractère**
DET_TYPE	CL_FNAME	.	CHAR
DET_TYPE	CL_LNAME	.	CHAR
DET_TYPE	ELAPSED	.	DURATION
DET_TYPE	ORDER_DATE		DATE

11. Attributs personnalisés (suite)

DET_TYPE	ELAPSED	.	DURATION
DET_TYPE	ORDER_DATE	.	DATE
DET_TYPE	ORDER_NO	.	INT
DET_TYPE	ORDER_TIME	.	TIME
DET_TYPE	PRICE	.	CURR
DET_TYPE	SAMPLING_W	.	DOUBLE
DET_TYPE	SHIPPED	.	TIMESTAMP
DET_TYPE	SHIP_STATUS	.	CHAR
ROLE	CL_FNAME	.	IDENT
ROLE	CL_LNAME	.	IDENT
ROLE	ELAPSED	.	CONTI
ROLE	ORDER_DATE	.	CATEG
ROLE	ORDER_NO	.	IDENT
ROLE	ORDER_TIME	.	CATEG
ROLE	PRICE	.	CONTI
ROLE	SAMPLING_W	.	WEIGHT
ROLE	SHIPPED	.	CATEG

12. Détruire des fichiers

- **DELETE** sert à supprimer des fichiers d'une bibliothèque :

```
PROC DATASETS  LIBRARY=WH  NOLIST;
  DELETE CLIENTS  SALES2023  DEPT;
QUIT;
```

- Une note au journal indique les fichiers détruits:

```
26
27          PROC DATASETS  LIBRARY=WH  NOLIST;
28            DELETE CLIENTS  SALES2023  DEPT;
29          QUIT;

NOTE: Deleting WH.CLIENTS (memtype=DATA).
NOTE: Deleting WH.SALES2023 (memtype=DATA).
NOTE: Deleting WH.DEPT (memtype=DATA).
NOTE: PROCEDURE DATASETS a utilisé (Durée totale du trait
```

12. Détruire des fichiers (suite)

- Les listes de fichiers sont permises:

```
DELETE SALES2019-SALES2023  CLI:;
```

- Détruire d'autres fichiers que des tables/vues exige l'emploi de **MEMTYPE=** (ou **MT=**) pour indiquer le type de fichier
- Exemple – détruire un catalogue et un gabarit :

```
PROC DATASETS  LIBRARY=WH  NOLIST;
  DELETE FORMATS(MT=CATALOG)
         MY_TEMPLATES(MT=ITEMSTOR);
QUIT;
```

12. Détruire des fichiers (suite)

- Impossible de préfixer les fichiers par leur bibliothèque – l'opération vise uniquement la bibliothèque par défaut

- Remplacer **DELETE** par **SAVE** pour détruire tous les fichiers d'une bibliothèque sauf ceux nommés

- Pour vider complètement une bibliothèque:

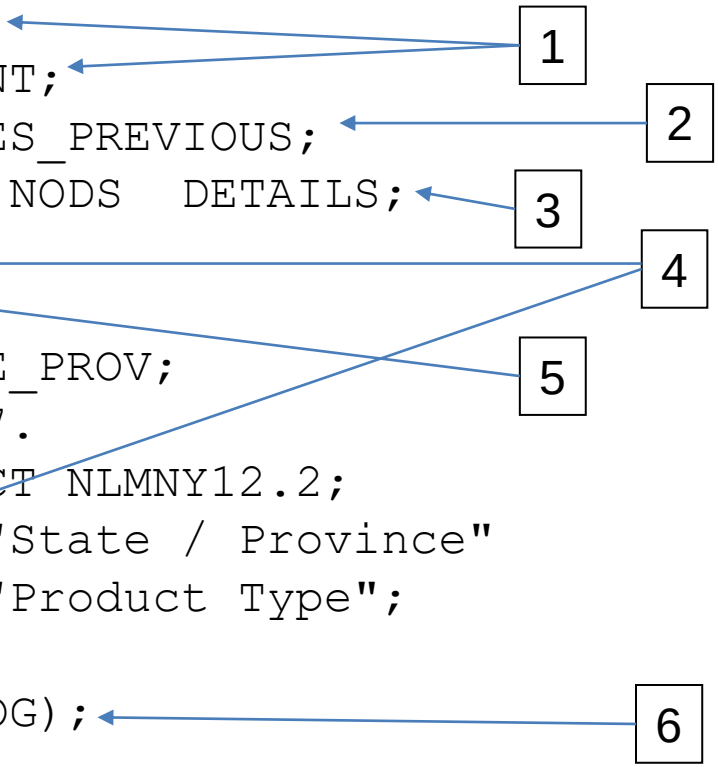
```
PROC DATASETS  LIBRARY=WH  KILL;
QUIT;
```

13. Autres tâches

- Plusieurs autres tâches possibles avec **PROC DATASETS** :
  - Indexer des tables
  - Gérer des tables « en générations »
  - Usage des fichiers d'audit
  - Définir des contraintes d'intégrité

14. Exemple intégratif

```
PROC DATASETS  LIBRARY=WH  NOLIST;
  COPY  IN=WORK  OUT=WH;
    SELECT  SALES_CURRENT;
  CHANGE  SALES2022=SALES_PREVIOUS;
  CONTENTS  DATA=_ALL_  NODS  DETAILS;
  RUN;
  MODIFY SALES_CURRENT;
    RENAME STATE = STATE_PROV;
    FORMAT PERIOD YYMMP7.
           ACTUAL PREDICT NLMNY12.2;
    LABEL STATE_PROV = "State / Province"
          PRODTYPE   = "Product Type";
  RUN;
  DELETE FOR: (MT=CATALOG);
QUIT;
```



14. Exemple intégratif (suite)

1. Copie table *WORK.SALES_CURRENT* dans bibliothèque *WH*
2. Renomme table *SALES2022* qui devient *SALES_PREVIOUS*
3. Affiche le contenu de la bibliothèque
4. Exécute les énoncés précédents, en un bloc, si pas d'erreur
5. Modifie les attributs de la table *SALES_CURRENT*
6. Détruit les catalogues SAS dont le nom débute par *FOR*

14. Exemple intégratif (suite)

- Plusieurs autres tâches possibles avec **PROC DATASETS** :
  - Indexer des tables
  - Gérer des tables « en générations »
  - Usage des fichiers d'audit
  - Définir des contraintes d'intégrité

Conclusion

- Procédure **DATASETS** compte des outils fort utiles et efficaces pour gérer les fichiers SAS
- Peut remplacer avantageusement l'étape **DATA** dans nombre de situations
- Alternative parfois plus flexible et efficace que **PROC SQL**

POUR PLUS D'INFORMATIONS

Jean Hardy
Services Conseils Hardy Inc.
418-626-1666
jhardy@schardy.qc.ca
Web: www.schardy.qc.ca

- Merci à Aubin Kouakou et Alain Voyer pour leurs judicieux commentaires