STT1682 – Progiciels en Statistique et Actuariat
Cours 8 – PROC SQL

Intro

Le language SQL (Structured Query Language) est un language utilisé à travers plusieurs applications pour extraire/modifier de l'information entre des bases/tables relationelles.

SAS a créé une procédure appelé le PROC SQL très puissante utilisant une syntaxe et une logique similaire au language SQL, mais adapté au language SAS. À travers un PROC SQL, nous serons capable de repliquer la majorité des traitements et procédures appris jusqu'à présent dans ce cours.

Syntaxe

Étant donné la quantité élevé de traitement possible avec un PROC SQL, nous verrons une syntaxe simplifié à travers ce cour :

```
PROC SQL;
CREATE TABLE BASESORTANTE AS
SELECT VAR1 as VAR1_new format=..., VAR2 as VAR2_new  format=..., ...
FROM BASEENTRANTE
WHERE CONDITIONS
GROUP BY VAR1, VAR2,...
ORDER BY VAR1, VAR2,...;
QUIT;
```

Note

- L'ordre des déclarations est importante.
- Selon vos besoins, plusieurs des étapes peuvent être tout simplement enlevés. Seulement le SELECT et FROM sont obligatoires.
- À noter que des virgules seront utilisés pour séparés les différentes variables

CREATE TABLE

-Créer la table sortante, similaire à la déclaration DATA

SELECT

- À cette étape, nous devons lister les variables que nous désirons garder dans la base de donnée, similaire à l'option KEEP.
- À noter que * peut être utilisé pour garder toutes les variables.
- L'option "as" peut être utilisé pour renommer une variable **(optionel)**
- Un format peut être créé pour chacune des variables en ajoutant format=... suivant la variable.
- Toutes nouvelles variables peut être créé à cette étape, il suffit d'écrire l'opération SAS pour la définir et lui définir un nom suivant le "as"

FROM
-Spécifie le nom de la (ou des) bases de données entrante en lecture, similaire à la déclaration SET

WHERE
Étape pour effectuer des filtres à la lecture, similaire à l'option WHERE

GROUP BY
Étape pour spécifier le regroupement de variable par lequel on veut sommariser la base, similaire à la déclaration CLASS dans un PROC SUMMARY

ORDER BY
Étape pour spécifier le trie voulu dans la base de donnée sortante, similaire à la déclaration BY d'un PROC SORT. Pour trier de façon décroissante, la variable devra être suivit de l'option "desc"

Exemple #1 – Création de Base

```
DATA BASE1;
x=1;y=1;OUTPUT;
x=2;y=1;OUTPUT;
x=3;y=1;OUTPUT;
RUN;

PROC SQL;
CREATE TABLE EXEMPLE1 AS
SELECT *, (x+y) as z format=8.1
FROM BASE1
WHERE x<3
ORDER BY X desc;
QUIT;
```

Sommarisation de Base

À l'aide d'un PROC SQL, il sera possible de sommariser plusieurs observations des variables et d'extraire diverses variables statistiques (somme,maximum...) similaire à ce que faisait un PROC SUMMARY .

Cependant, une différence majeure est qu'avec un PROC SQL, nous pouvons aussi calculer ces variables statistiques sommaires sans nécessairement sommariser la base de donnée.

Cela implique que nous pourrons utiliser ces variables statistiques calculés à partir de plusieurs observations avec les valeurs de chacune de ces observations.

Par exemple, en un seul PROC SQL, il sera possible de calculer pour chacune des observations le ratio de VAR1 par rapport à la somme de toutes les valeurs VAR1, ce qui n'était précedemment pas possible à faire en une seule étape.

Les mêmes variables statistiques que celles utilisés dans le PROC SUMMARY pourront être utilisés à l'intérieur d'un PROC SQL soit : SUM(), MEAN(), MAX(), MIN(), N(), STD()...

Exemple #2 – Sommarisation de Base

```
PROC SQL;
CREATE TABLE EXEMPLE2 AS
SELECT SUM(X) AS SUM_X
FROM BASE1
GROUP BY Y;
QUIT;
```

Exemple #3 – Variable Statistique sans Sommarisation

```
PROC SQL;
CREATE TABLE EXEMPLE3 AS
SELECT *, SUM(X) AS SUM_X
FROM BASE1;
QUIT;
```

Liaison de Base de Données

Il est aussi possible de lier plusieurs bases de données entre-elles à l'aide d'un PROC SORT. Le PROC SORT va aussi permettre des conditions de liaisons beaucoup plus complexes que celles utilisés avec la déclaration MERGE ou l'option INDEX.

Liaison par défaut

Par défaut le PROC SQL va tenter de faire un produit cartésien des différentes tables. En d'autres mots, il va produire toutes les combinaisons différentes possibles de chacune des variables des différentes bases de données.

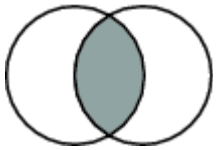
Exemple #4 – Liaison de Base

```
DATA BASE1;
x=1;y=2;OUTPUT;
x=2;y=3;OUTPUT;
RUN;

DATA BASE2;
x=2;z=5;OUTPUT;
x=3;z=6;OUTPUT;
x=4;z=9;OUTPUT;
RUN;

PROC SQL;
CREATE TABLE EXEMPLE4 AS
SELECT *
FROM BASE1,BASE2;
QUIT;
```

INNER JOIN
À l'aide de la déclaration WHERE, il sera possible de spécifier un filtre afin d'effectuer ce qu'on appele un INNER JOI où le PROC SQL va seulement conserver les observations qu'il a réussit à lier dans les deux bases de données. Voici l'information conservé représenté par le diagramme de Venne ci-dessous :

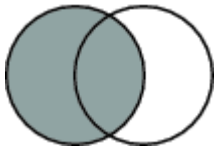


ATTENTION, si la clé n'est pas unique, l'information va être dédoublé

Exemple #5 – Liaison de Base (INNER JOIN)

```
PROC SQL;
CREATE TABLE EXEMPLE5 AS
SELECT *
FROM BASE1,BASE2
WHERE BASE1.X=BASE2.X;/*A Cette etape on force SAS a conserver seulement les observations ou
x de BASE1 est egale a X de BASE2 */
QUIT;
```

LEFT JOIN
Si à la place, on désire conserver toutes l'information de BASE1 et seulement ajouter l'information qu'on a réussit à lier de BASE2, on va effectuer un LEFT JOIN représenté par le graphique ci-dessous :



Exemple #6 – Liaison de Base (LEFT JOIN)

```
PROC SQL;
CREATE TABLE EXEMPLE6 AS
SELECT *
FROM BASE1 LEFT JOIN BASE2
ON BASE1.X=BASE2.X;*ON devra etre utiliser au lieu du WHERE;
QUIT;
```

Note : Le RIGHT JOIN existe aussi et produira le résultat inverse

Liaisons Complexes
Finalement, il sera aussi possible de lier des bases de données avec des conditions de liaisons plus complexes tel que des inégalités. Tout cela sera traité à travers la déclaration WHERE ou ON dans le cas des LEFT/RIGHT join.

Exemple #7 – Liaison de Base (Complexe)

```
DATA BASE1;
x=1;y=10;OUTPUT;
x=11;y=20;OUTPUT;
x=21;y=30;OUTPUT;
RUN;

DATA BASE2;
x=1;z=5;OUTPUT;
x=11;z=5;OUTPUT;
RUN;

PROC SQL;
CREATE TABLE EXEMPLE7 AS
SELECT *
FROM BASE1 LEFT JOIN BASE2
ON BASE1.X=BASE2.X
AND BASE2.Z LE BASE1.Y
AND BASE2.Z GE BASE1.X;
QUIT;
```