Power Week

Université IBM i 2019

22 et 23 mai

IBM Client Center Paris

S25 – Les fonctions OLAP de SQL

Marie Gris
Volubis
mgris@volubis.fr

Power Week

Université IBM i 2019

22 et 23 mai

IBM Client Center Paris

i for Business

Volubis.fr

Conseil et formation sur OS/400, I5/OS puis IBM *i* depuis 1994 !

Dans nos locaux, vos locaux ou par Internet

Les fonctions OLAP de SQL

- Fonctions d'agrégation VS fonctions scalaires
  - Fonctions d'agrégation historiques
  - Focus sur le COUNT
- Fonctions de la clause Group By (rollup, grouping sets…)
- Fonctions de classement
- Fonctions statistiques

Les fonctions OLAP de SQL

- Fonctions d'agrégation VS fonctions scalaires
  - Fonction d'agrégation historiques
    - Sans commentaires :
      - Sum
      - Avg
      - Count
      - Min
      - Max
      - VAR → la variance
        - plus elle est élevée, plus les valeurs sont hétérogènes
        - plus elle est faible, plus elles sont homogènes
      - STDDEV → écart type (moyenne des écarts)
        - c'est la racine carrée de la variance

Les fonctions OLAP de SQL

- Fonctions d'agrégation VS fonctions scalaires

  - Certaines fonctions sont scalaires ET d'agrégation :
    - Min(prix), donne le plus petit prix (Agrégation)
    - Min(priha , privente) donne le plus petit des deux prix (Scalaire)

  - Pareil pour la fonction max
    - Max(datcde, datliv) donne la plus grande des deux dates

Les fonctions OLAP de SQL

- Fonctions d'agrégation VS fonctions scalaires
  - Toutes les fonctions d'agrégation travaillent avec des ensembles

    - Cet ensemble peux être le jeux d'enregistrement complet (la table)
      - Select Min(prix) from ARTICLES,
        donne le plus petit prix du fichier articles

      - Select count(*) from ARTICLES,
        donne le nombre d'articles
        Plus exactement : le nombre de lignes extraites

Les fonctions OLAP de SQL

- Fonctions d'agrégation VS fonctions scalaires
  - Count
    - Comptage, que compte-t-on ?
      Le nombre de lignes extraites, on l'a vu.
    - Select count(*) from CLIENTS → combien de clients
    - Select count(*) from CLIENTS where depcli = 75
      → combien de clients parisiens

Les fonctions OLAP de SQL

- Fonctions d'agrégation VS fonctions scalaires
  - Count
    - Que représente une ligne ?
      - je dois compter le nombre de commandes passées par des clients parisiens, cette année (le département est dans le fichier clients)

    - Select count(*) from CLIENTS left outer join COMMANDES
      using(nocli) where (year(datcde) = year(now()) or datcde is null)
      and depcli = 75

      je compte le nombre de lignes,
      est-ce le nombre de commandes ?
    - **NON**, un client sans commande = une ligne (left outer)

Les fonctions OLAP de SQL

- Fonctions d'agrégation VS fonctions scalaires
  - Count sur une colonne
    - Que représente une ligne ?
      - je dois compter le nombre de commandes passées par des clients parisiens, cette année (le département est dans le fichier clients)

    - Select count(nocde) from CLIENTS left outer join COMMANDES using(nocli) using(nocli) where (year(datcde) = year(now()) or datcde is null) and depcli = 75

    je compte le nombre de lignes où nocde est non null,
    est-ce le nombre de commandes ?
    - **Oui**

Les fonctions OLAP de SQL

- Fonctions d'agrégation VS fonctions scalaires
  - Count
    - Que représente une ligne ?
      - je dois compter le nombre de clients à livrer la semaine prochaine

    - Select count(*) from CLIENTS join COMMANDES using(nocli)
      where datcde between next_day(current date, 'DIM') + 1 day
      AND next_day(current date, 'DIM') + 7 days

      je compte le nombre de lignes,
      est-ce le nombre de clients ?
    - **Non,** le nombre de commandes !

Les fonctions OLAP de SQL

- Fonctions d'agrégation VS fonctions scalaires
  - Count
    - Que représente une ligne ?
      - je dois compter le nombre de clients à livrer la semaine prochaine

    - Select count(distinct nocli) from CLIENTS join COMMANDES using(nocli)
      where datcde between next_day(current date, 'DIM') + 1 day
      AND next_day(current date, 'DIM') + 7 days

      je compte le nombre de nocli différents,
      est-ce le nombre de clients ?
      Oui Par défaut count(expression) est équivalent à count(ALL expression),
      Count(DISTINCT expression) permet d'éliminer les doublons et les NULL

Les fonctions OLAP de SQL

- Fonctions de la clause Group By (rollup, grouping sets…)
  - Toutes les fonctions d'agrégation travaillent avec des ensembles
    - Cet ensemble peux être matérialisé par GROUP BY
      - De nombreux exemple « SQL as a service » utilisent GROUP BY
      - Les mêmes services sont aujourd'hui intégrés à DB2 Web Query
  - La fonction d'agrégation s'applique alors à un groupe de lignes et plus à toute une table
    - Lorsqu'une requête s'exécute avec GROUP BY la BDD répartit d'abord les lignes de l'ensemble dans les différents groupes
    - C'est ensuite que s'applique la fonction sur chacun des groupes (sous-ensemble)

Les fonctions OLAP de SQL

- Fonctions de la clause Group By (rollup, grouping sets…)
  - Si vous avez besoin d'afficher une information non déterminante mettez la aussi dans le Group By

```
select pr_code, pr_nom, count(*) as nombre
 from vins join producteurs using(pr_code) group by pr_code
```

```
Mon Jan 02 15:09:31 CET 2017 ]  Run Selected...
select pr_code, pr_nom, count(*) as nombre from vins join producteurs using(pr_cod
SQL State: 42803
endor Code: -122
lessage: [SQL0122] Colonne PR_NOM ou expression de la liste SELECT est incorrecte.
```

- OK

```
select pr_code, pr_nom, count(*) as nombre
 from vins join producteurs using(pr_code) group by pr_code , pr_nom
```

_CODE	PR_NOM	NOMBRE
5611	Tulbagh Cooperative	1
4082	Azienda Agricola Luisa Eddi	6
2037	Domaine J.-L.Fougeray	5
1421	Le Clos des Mothèles	2

Les fonctions OLAP de SQL

- Fonctions de la clause Group By (rollup, grouping sets…)
  - Si vous avez besoin d'afficher une information non déterminante plutôt que regrouper sur 2 colonnes et pour utiliser un index de tri plus court
    - Vous pouvez compenser l'absence d'une colonne dans le group by en la « camouflant » dans une fonction d'agrégation

```
Select pr_code, max(pr_nom) as nom , Count(*) As nombre
   From vins
   Join producteurs Using(pr_code)
   Group By pr_code
```

PR_CODE	NOM	NOMBRE
1	Château Marquis de Terme	1
2	Château Maucaillou	3
3	Château Mayne Lalande	2
4	Château Monbrison	1
5	Château Moulin La Pitié	1
6	Château Moulin Saint-Georges	1
7	Château Canon-Moueix	1
8	Château Duhart-Milon	1

  - OK

Les fonctions OLAP de SQL

- Fonctions de la clause Group By (rollup, grouping sets…)
  - Si vous avez besoin d'une sélection sur une agrégation, utilisez HAVING
    - HAVING permet un filtre sur le résultat d'un GROUP BY, il doit donc être placé après ce regroupement

```
Select pr_code, Max(pr_nom) As nom , Count(*) As nombre
  From vins
  Join producteurs Using(pr_code)
  Group By pr_code
  Having Count(*) > 9
```

PR_CODE	NOM	NOMBRE
116	Château de Haux	11
281	Domaine Vincent Prunier	12
283	Domaine Raphet Jean et fils	10
287	Domaine Billaud-Simon	12
297	Château de Fuissé	10
300	Domaine Jean-Marc Brocard	11

Les fonctions OLAP de SQL

- Fonctions de la clause Group By (rollup, grouping sets…)
  - Les fonctions d'agrégation ne peuvent pas être imbriquées
    - AVG(COUNT(*)) n'est pas valide

```
Select pr_code, Max(pr_nom) As nom , avg(Count(*)) As nombre
   From vins
   Join producteurs Using(pr_code)
   Group By pr_code
   Having Count(*) > 9
```

```
Thu Apr 04 15:30:13 CEST 2019 ]  Exécution de la sélection...
Select pr_code, Max(pr_nom) As nom , avg(Count(*)) As nombre From vins Join producte
Etat SQL : 42607
ode fournisseur : -112
essage : [SQL0112] L'argument de la fonction AVG contient une autre fonction. Cause . . . .
agrégation ou une spécification OLAP. Seules les expressions sans fonction d'agrégation ou s
LAP. Que faire . . . : Corrigez l'appel de fonction. Renouvelez votre demande.

Instructions ayant échoué : 1
```

Les fonctions OLAP de SQL

- Fonctions de la clause Group By (rollup, grouping sets…)
  - Les fonctions d'agrégation ne peuvent pas être imbriquées
    - Utilisez plutôt

```
1 Select Avg(nombre)
2    From (Select pr_code, Max(pr_nom) As nom , Count(*) As nombre
3          From vins
4          Join producteurs Using(pr_code)
5          Group By pr_code
6          Having Count(*) > 9)
7    As temp
```

00001
12

Les fonctions OLAP de SQL

- Fonctions de la clause Group By (rollup, grouping sets…)
  - Les fonctions d'agrégation ne peuvent pas être imbriquées
    - Elles supportent le DISTINCT
      - Notez la différence entre ces 2 moyennes

```
19 Select Avg( nombre)
20    From (Select pr_code, Max(pr_nom) As nom , Count(*) As nombre
21          From vins
22          Join producteurs Using(pr_code)
23          Group By pr_code
24          Having Count(*) > 9);
```

```
00001
   12
```

```
19 Select Avg(distinct nombre)
20    From (Select pr_code, Max(pr_nom) As nom , Count(*) As nombre
21          From vins
22          Join producteurs Using(pr_code)
23          Group By pr_code
24          Having Count(*) > 9);
```

```
00001
   16
```

Les fonctions OLAP de SQL

- Fonctions de la clause Group By (rollup, grouping sets…)
  - Les fonctions d'agrégation ne peuvent pas être imbriquées
    - Ou utilisez les CTE

```
With temp
   As (Select pr_code, Max(pr_nom) As nom , Count(*) As nombre
         From vins
         Join producteurs Using(pr_code)
         Group By pr_code
         Having Count(*) > 9)
   Select Avg(nombre)
      From temp
```

```
0001
   12
```

    - Elles vous permettent ceci

```
With temp
   As (Select pr_code, Max(pr_nom) As nom , Count(*) As nombre
         From vins
         Join producteurs Using(pr_code)
         Group By pr_code
         Having Count(*) > 9)
   Select *
      From temp
      Where nombre > (Select Avg(nombre)
                      From temp)
```

R_CODE	NOM	NOMBRE
301	Domaine Patrick Javillier	13
310	Cave des Vignerons de Buxy	15
327	Domaine François Gay	13
357	Fontaine-Gagnard	16
358	Fernand & Laurent Pillot	18

Les fonctions OLAP de SQL

- Fonctions de la clause Group By (rollup, grouping sets…)
  - Un regroupement peut s'effectuer sur une expression
    - Exemple avec le fichier ou nous historisons les accès à notre site Volubis.fr

```
12 Select Count(*) As nombre
13    From httplog.httplog;
14
```

NOMBRE
59810754

```
12 Select Year(logtime) As annee, Count(*) As nombre
13    From httplog.httplog
14    Group By Year(logtime);
15
```

ANNEE	NOMBRE
2012	3903384
2013	6245919
2014	6815648
2015	7895074
2016	12080356
2017	10162076
2018	9630640
2019	3077657

Les fonctions OLAP de SQL

- Fonctions de la clause Group By (rollup, grouping sets…)
  - Un groupe peut être identifié par plusieurs colonnes
    - Si l'une change, le groupement change

```
12 Select Year(logtime) As annee, month(logtime) as mois, Count(*) As nombre
13    From httplog.httplog
14    Group By Year(logtime), month(logtime);
15
```

ANNEE	MOIS	NOMBRE
2017	10	693521
2017	11	819627
2017	12	405546
2018	1	698889
2018	2	545378
2018	3	806988
2018	4	747876
2018	5	592672
2018	6	696867
2018	7	691327
2018	8	572279
2018	9	759278
2018	10	852639

Les fonctions OLAP de SQL

- Fonctions de la clause Group By (rollup, grouping sets…)
  - Le tri n'est pas implicite avec SQE, il faut trier explicitement

```
11
12 Select Year(logtime) As annee, Month(logtime) As mois, Count(*) As nombre
13    From httplog.httplog
14    Group By Year(logtime), Month(logtime)
15    Order By annee, mois;
16
```

ANNEE	MOIS	NOMBRE
2012	1	356076
2012	2	342810
2012	3	371224
2012	4	329299
2012	5	295196
2012	6	327293
2012	7	255034
2012	8	204636
2012	9	320999
2012	10	376621
2012	11	381707
2012	12	342489
2013	1	439228

Les fonctions OLAP de SQL

- Fonctions de la clause Group By (rollup, grouping sets…)
  - Permet une analyse multi-dimensionnelle
  - 2 fonctions particulières
    - ROLLUP génère un jeu de résultat pour les colonnes sélectionnées
    - CUBE Envisage toutes les combinaisons possibles pour les colonnes sélectionnées, y compris les totaux partiels et intermédiaires
  - GROUPING SET Permet de combiner les différents types de regroupements et de les hiérarchiser

Les fonctions OLAP de SQL

- Fonctions de la clause Group By (rollup, grouping sets…)
  - Group by ROLLUP permet des sous-totaux
    - Le group by produit un résumé par ligne et le ROLLUP ajoute des résumés pour les autres niveaux

```
2 Select Year(logtime) As annee, Month(logtime) As mois,
3 Count(*) As nombre
4    From httplog.httplog
5    Group By rollup(Year(logtime), Month(logtime))
6    Order By annee, mois;
```

ANNEE	MOIS	NOMBRE
2012	1	356076
2012	2	342810
2012	3	371224
2012	4	329299
2012	5	295196
2012	6	327293
2012	7	255034
2012	8	204636
2012	9	320999
2012	10	376621
2012	11	381707
2012	12	342489
2012	-	3903384

erminé : 97 lignes extraites

    - Et un total général

2019	4	20730
2019	-	3077657
-	-	59810754

miné : 97 lignes extraites

Les fonctions OLAP de SQL

- Fonctions de la clause Group By (rollup, grouping sets…)
  - ROLLUP ajoute des résumés pour le niveau choisi

```
Select Year(logtime) As annee, Month(logtime) As mois,
Count(*) As nombre
   From httplog.httplog
   Group By Month(logtime),  rollup(Year(logtime))
   Order By annee, mois;
```

NNEE	MOIS	NOMBRE
2019	1	931458
2019	2	1032871
2019	3	1092598
2019	4	20730
-	1	6028689
-	2	5544978
-	3	6256421
-	4	4653408
-	5	3982029
-	6	4126446
-	7	3873403
-	8	3723414
-	9	4400116

  - Ici sans total général

-	11	5431650
-	12	6819837

Terminé : 100 lignes extraites

Messages Variables globales e

Les fonctions OLAP de SQL

- Fonctions de la clause Group By (rollup, grouping sets…)
  - GROUPING SETS produit un résumé pour chaque niveau indiqué

```
12 Select Year(logtime) As annee, Month(logtime) As mois,
13 Count(*) As nombre
14    From httplog.httplog
15    Group By   grouping sets(Month(logtime),Year(logtime))
16    Order By annee, mois;
17
```

ANNEE	MOIS	NOMBRE
2015	-	7895074
2016	-	12080356
2017	-	10162076
2018	-	9630640
2019	-	3077657
-	1	6028689
-	2	5544978
-	3	6256421
-	4	4653408
-	5	3982029
-	6	4126446
-	7	3873403
-	8	3723414

Terminé : 20 lignes extraites

  - Pour chaque année
  - Pour chaque mois

(toutes années confondues)

Les fonctions OLAP de SQL

- Fonctions de la clause Group By (rollup, grouping sets…)
  - CUBE produit un résumé pour chaque niveau plus un total général

```
12 Select Year(logtime) As annee, Month(logtime) As mois,
13 virtualhost, Count(*) As nombre
14    From httplog.httplog
15    Group By cube( virtualhost, Month(logtime),Year(logtime))
16    Order By annee, mois;
17
18
```

ANNEE	MOIS	VIRTUALHOST	NOMBRE
-	11	www.commonfrance.fr	4029
-	11	www.cibretagne.org	43253
-	11	-	5431650
-	12	www.cibretagne.org	42815
-	12	www.clubas400pl.org	45693
-	12	www.volubis.fr	6725815
-	12	www.commonfrance.fr	5514
-	12	-	6819837
-	-	www.clubas400pl.org	694478
-	-	www.commonfrance.fr	62915
-	-	www.cibretagne.org	770088
-	-	www.volubis.fr	58283273
-	-	-	59810754

Terminé : 464 lignes extraites

ANNEE	MOIS	VIRTUALHOST	NOMBRE
2018	12	www.cibretagne.org	22958
2018	12	www.clubas400pl.org	20896
2018	12	www.commonfrance.fr	5514
2018	12	www.volubis.fr	1738061
2018	12	-	1787429
2018	-	www.cibretagne.org	406789
2018	-	www.clubas400pl.org	269481
2018	-	www.commonfrance.fr	37965
2018	-	www.volubis.fr	8916405
2018	-	-	9630640
2019	1	www.cibretagne.org	26753
2019	1	www.clubas400pl.org	24966
2019	1	www.commonfrance.fr	7901

Les fonctions OLAP de SQL

- Fonctions de la clause Group By (rollup, grouping sets…)
  - Grouping() permet de savoir si vous êtes au niveau détail ou total de la colonne indiquée

```
Select Year(logtime) As annee, Month(logtime) As mois,
virtualhost, Count(*) As nombre,
grouping(Year(logtime)) as total_annee,
grouping(month(logtime)) as total_mois,
grouping(virtualhost) as total_host
   From httplog.httplog
   Group By cube( virtualhost, Month(logtime),Year(logtime))
   Order By annee, mois;
```

ANNEE	MOIS	VIRTUALHOST	NOMBRE	TOTAL_ANNEE	TOTAL_MOIS	TOTAL_HOST
2012	1	www.cibretagne.org	4262	0	0	0
2012	1	www.clubas400pl.org	3695	0	0	0
2012	1	www.volubis.fr	348119	0	0	0
2012	1	-	356076	0	0	1
2012	2	www.cibretagne.org	3816	0	0	0
2012	2	www.clubas400pl.org	4292	0	0	0
2012	2	www.volubis.fr	334702	0	0	0
2012	2	-	342810	0	0	1
2012	3	www.cibretagne.org	3676	0	0	0
2012	3	www.clubas400pl.org	6154	0	0	0
2012	3	www.volubis.fr	361394	0	0	0

ANNEE	MOIS	VIRTUALHOST	NOMBRE	TOTAL_ANNEE	TOTAL_MOIS	TOTAL_HOST
2016	12	www.volubis.fr	2269185	0	0	0
2016	12	-	2278753	0	0	1
2016	-	www.cibretagne.org	44725	0	1	0
2016	-	www.clubas400pl.org	63284	0	1	0
2016	-	www.volubis.fr	11972347	0	1	0
2016	-	-	12080356	0	1	1
2017	1	www.cibretagne.org	4663	0	0	0
2017	1	www.clubas400pl.org	6644	0	0	0

ANNEE	MOIS	VIRTUALHOST	NOMBRE	TOTAL_ANNEE	TOTAL_MOIS	TOTAL_HOST
-	11	-	5431650	1	0	1
-	12	www.cibretagne.org	42815	1	0	0
-	12	www.clubas400pl.org	45693	1	0	0
-	12	www.commonfrance.fr	5514	1	0	0
-	12	www.volubis.fr	6725815	1	0	0
-	12	-	6819837	1	0	1
-	-	www.cibretagne.org	770088	1	1	0
-	-	www.clubas400pl.org	694478	1	1	0
-	-	www.commonfrance.fr	62915	1	1	0
-	-	www.volubis.fr	58283273	1	1	0
-	-	-	59810754	1	1	1

Les fonctions OLAP de SQL

- Fonctions de la clause Group By (rollup, grouping sets…)
  - Grouping() permet donc, de placer tous les totaux en fin de liste

```
2 Select Year(logtime) As annee, Month(logtime) As mois, Count(*) As nombre, grouping(Year(logtime))
3    As total_annee
4    From httplog.httplog
5    Group By cube(virtualhost, Month(logtime),Year(logtime))
6    Order By total_annee;
7
8
```

ANNEE	MOIS	NOMBRE	TOTAL_ANNEE
2019	-	2900214	0
2019	-	3077657	0
-	1	108352	1
-	2	88313	1
-	3	93878	1
-	4	57393	1
-	5	50565	1
-	6	58371	1
-	7	48527	1
-	8	48061	1
-	9	59405	1
-	10	71155	1
-	11	43253	1

erminé : 464 lignes extraites

Les fonctions OLAP de SQL

- Fonctions de classement
  - Fonctions OLAP livrées depuis la version …. 6.1
    - ROW_NUMBER() OVER ( )
    - RANK() OVER ( )
    - DENSE_RANK() OVER ( )

Les fonctions OLAP de SQL

- Fonctions de classement
  - ROW_NUMBER() OVER ( )
    - Numérotation de chaque ligne

```
17
18    Select vin_nom, cav_prxactuel, row_number()
19    over() as num_rang
20    From vins
21    Join ma_cave Using(vin_code);
22
```

VIN_NOM	CAV_PRXACTUEL	NUM_RANG
Jurançon Noblesse du Temps	34.51	1
Jurançon L'Eminence	41.03	2
Madiran Montus	29.53	3
Saint-Joseph	23.35	4
Saint-Joseph	12.43	5
Condrieu	23.35	6
Condrieu Coteaux de Chéry	35.70	7
Saumur-Champigny Vieilles vignes	16.14	8
Château Margaux	233.98	9
Irouléguy	15.89	10
Rioja Santiago	16.06	11
Muscadet Sèvre-et-Maine Guy Bossard	5.45	12
Château d'Yquem	158.60	13
Nebbiolo d'Alba Bric Merli	23.46	14

Les fonctions OLAP de SQL

- Fonctions de classement
  - ROW_NUMBER() OVER ( )
    - Numérotation de chaque ligne selon un critère de tri indiqué par OVER()

```
18  Select vin_nom, cav_prxactuel, row_number()
19  over(order by cav_prxactuel) as num_rang
20  From vins
21  Join ma_cave Using(vin_code);
22
```

VIN_NOM	CAV_PRXACTUEL	NUM_RANG
Muscadet Sèvre-et-Maine Guy Bossard	5.45	1
Saint-Joseph	12.43	2
Trouléguy	15.89	3
Rioja Santiago	16.06	4
Saumur-Champigny Vieilles vignes	16.14	5
Saint-Joseph	23.35	6
Condrieu	23.35	7
Nebbiolo d'Alba Bric Merli	23.46	8
Madiran Montus	29.53	9
Jurançon Noblesse du Temps	34.51	10
Condrieu Coteaux de Chéry	35.70	11
Jurançon L'Eminence	41.03	12
Château d'Yquem	158.60	13
Château Margaux	233.98	14

Les fonctions OLAP de SQL

- Fonctions de classement
  - ROW_NUMBER() OVER ( )
    - Numérotation de chaque ligne selon un critère de tri descendant

```
18  Select vin_nom, cav_prxactuel, row_number()
19  over(order by cav_prxactuel desc) as num_rang
20  From vins
21  Join ma_cave Using(vin_code);
22
```

VIN_NOM	CAV_PRXACTUEL	NUM_RANG
Château Margaux	233.98	1
Château d'Yquem	158.60	2
Jurançon L'Eminence	41.03	3
Condrieu Coteaux de Chéry	35.70	4
Jurançon Noblesse du Temps	34.51	5
Madiran Montus	29.53	6
Nebbiolo d'Alba Bric Merli	23.46	7
Saint-Joseph	23.35	8
Condrieu	23.35	9
Saumur-Champigny Vieilles vignes	16.14	10
Rioja Santiago	16.06	11
Irouléguy	15.89	12
Saint-Joseph	12.43	13
Muscadet Sèvre-et-Maine Guy Bossard	5.45	14

Les fonctions OLAP de SQL

- Fonctions de classement
  - ROW_NUMBER() OVER ( )
    - Numérotation de chaque ligne selon un critère de tri descendant

      Notez les Ex-aequo

```
18  Select vin_nom, cav_prxactuel, row_number()
19  over(order by cav_prxactuel desc) as num_rang
20  From vins
21  Join ma_cave Using(vin_code);
22
```

VIN_NOM	CAV_PRXACTUEL	NUM_RANG
Château Margaux	233.98	1
Château d'Yquem	158.60	2
Jurançon L'Eminence	41.03	3
Condrieu Coteaux de Chéry	35.70	4
Jurançon Noblesse du Temps	34.51	5
Madiran Montus	29.53	6
Nebbiolo d'Alba Bric Merli	23.46	7
Saint-Joseph	23.35	8
Condrieu	23.35	9
Saumur-Champigny Vieilles vignes	16.14	10
Rioja Santiago	16.06	11
Irouléguy	15.89	12
Saint-Joseph	12.43	13
Muscadet Sèvre-et-Maine Guy Bossard	5.45	14

Les fonctions OLAP de SQL

- Fonctions de classement
  - RANK() OVER() permet de gérer les ex-aequo

```
18  Select vin_nom, cav_prxactuel, rank()
19  over(order by cav_prxactuel desc) as num_rang
20  From vins
21  Join ma_cave Using(vin_code);
22
```

VIN_NOM	CAV_PRXACTUEL	NUM_RANG
Château Margaux	233.98	1
Château d'Yquem	158.60	2
Jurançon L'Eminence	41.03	3
Condrieu Coteaux de Chéry	35.70	4
Jurançon Noblesse du Temps	34.51	5
Madiran Montus	29.53	6
Nebbiolo d'Alba Bric Merli	23.46	7
Saint-Joseph	23.35	8
Condrieu	23.35	8
Saumur-Champigny Vieilles vignes	16.14	10
Rioja Santiago	16.06	11
Crouléguy	15.89	12
Saint-Joseph	12.43	13
Muscadet Sèvre-et-Maine Guy Bossard	5.45	14

Terminé : 14 lignes extraites

Les fonctions OLAP de SQL

- Fonctions de classement
  - DENSE_RANK() OVER() permet de gérer les ex-aequo sans interruption dans la numérotation

```
17
18    Select vin_nom, cav_prxactuel, dense_rank()
19    over(order by cav_prxactuel desc) as num_rang
20    From vins
21    Join ma_cave Using(vin_code);
22
```

VIN_NOM	CAV_PRXACTUEL	NUM_RANG
Château Margaux	233.98	1
Château d'Yquem	158.60	2
Jurançon L'Eminence	41.03	3
Condrieu Coteaux de Chéry	35.70	4
Jurançon Noblesse du Temps	34.51	5
Madiran Montus	29.53	6
Nebbiolo d'Alba Bric Merli	23.46	7
Saint-Joseph	23.35	8
Condrieu	23.35	8
Saumur-Champigny Vieilles vignes	16.14	9
Rioja Santiago	16.06	10
Irouléguy	15.89	11
Saint-Joseph	12.43	12
Muscadet Sèvre-et-Maine Guy Bossard	5.45	13

Les fonctions OLAP de SQL

- Fonctions de classement
  - Ces deux dernières fonctions sont plus fiables que FETCH FIRST
    - La sélection des 8 premiers ne permet pas d'afficher les ex-aequo du rang 8

```
17
18    Select vin_nom, cav_prxactuel, dense_rank()
19    over(order by cav_prxactuel desc) as num_rang
20    From vins
21    Join ma_cave Using(vin_code) fetch first 8 rows only;
22
```

VIN_NOM	CAV_PRXACTUEL	NUM_RANG
Château Margaux	233.98	1
Château d'Yquem	158.60	2
Jurançon L'Eminence	41.03	3
Condrieu Coteaux de Chéry	35.70	4
Jurançon Noblesse du Temps	34.51	5
Madiran Montus	29.53	6
Nebbiolo d'Alba Bric Merli	23.46	7
Saint-Joseph	23.35	8

Les fonctions OLAP de SQL

- Fonctions de classement
  - Revenons à notre fichier des « hits » sur Volubis.fr classement des meilleurs années

```
11  With temp
12    As (Select Year(logtime) As annee, Count(*) As nombre
13          From httplog.httplog
14          Group By Year(logtime))
15    Select annee, nombre, row_number()
16       over(Order By nombre Desc)
17       From temp;
```

ANNEE	NOMBRE	00003
2016	12080356	1
2017	10162076	2
2018	9630640	3
2015	7895074	4
2014	6815648	5
2013	6245919	6
2012	3903384	7
2019	3077657	8

Les fonctions OLAP de SQL

- Fonctions de classement
  - Le critère de tri final peut être différent du critère de tri du rang

```
1  With temp
2    As (Select Year(logtime) As annee, Month(logtime) As mois, Count(*) As nombre From
3          httplog.httplog Group By Year(logtime), Month(logtime))
4      Select annee, nombre, row_number()
5        over(Order By nombre Desc)
6        From temp
7        Order By annee, mois;
```

ANNEE	NOMBRE	00003
2012	356076	79
2012	342810	80
2012	371224	78
2012	329299	82
2012	295196	85
2012	327293	83
2012	255034	86
2012	204636	87
2012	320999	84
2012	376621	77
2012	381707	76
2012	342489	81
2013	439228	74

erminé : 88 lignes extraites

Les fonctions OLAP de SQL

- Fonctions de classement
  - On peut aussi partitionner le rang (ici dans une année)

```
1  With temp
2    As (Select Year(logtime) As annee, Month(logtime) As mois, Count(*) As nombre From
3          httplog.httplog Group By Year(logtime), Month(logtime))
4       Select annee, mois, nombre, row_number()
5          over(partition by annee Order By nombre Desc)
6          From temp
7          Order By annee, mois;
```

ANNEE	MOIS	NOMBRE	00004
2012	1	356076	4
2012	2	342810	5
2012	3	371224	3
2012	4	329299	7
2012	5	295196	10
2012	6	327293	8
2012	7	255034	11
2012	8	204636	12
2012	9	320999	9
2012	10	376621	2
2012	11	381707	1
2012	12	342489	6
2013	1	439228	12

Les fonctions OLAP de SQL

- Fonctions de classement
  - On peut aussi partitionner le rang (ici dans une année) ce qui permet de déduire des tendances. Ce n'est pas toujours le même mois le meilleur

```
11  With temp
12    As (Select Year(logtime) As annee, Month(logtime) As mois, Count(*) As nombre From
13          httplog.httplog Group By Year(logtime), Month(logtime))
14       Select annee, mois, nombre, row_number()
15          over(partition by annee Order By nombre Desc) as rang
16          From temp
17          Order By rang;
```

ANNEE	MOIS	NOMBRE	RANG
2012	11	381707	1
2013	12	696147	1
2014	3	643833	1
2015	11	738899	1
2016	12	2278753	1
2017	1	1615081	1
2018	12	1787429	1
2019	3	1092598	1
2012	10	376621	2
2013	11	606700	2
2014	4	604335	2
2015	12	734987	2
2016	11	1436302	2

Les fonctions OLAP de SQL

- Fonctions statistiques avec la 7.3
  - COVARIANCE(x, y)
  - Si la variance permet d'étudier les variations d'une variable par rapport à elle-même, la covariance va permettre d'étudier les variations simultanées de deux variables par rapport à leur moyenne respective.

  - Du résultat obtenu par cette mesure on en déduit que :
    - plus la covariance est faible et plus les séries sont indépendantes
    - plus elle est élevée et plus les séries sont liées (Elles varient ensemble)

Les fonctions OLAP de SQL

- Fonctions statistiques avec la 7.3
  - COVARIANCE_SAM
    - Comme la COVARIANCE, mais non biaisée (sur une population de n-1)
  - CORRELATION(x , y)
    - COVARIANCE divisée par le produit des écart-type
    - compris entre -1 et 1
      - Proche de 0, x et y évoluent de manière indépendante (non liée)
      - Proche de 1, ils évoluent de manière liée
        - ex : température extérieure et consommation de crèmes glacées
      - Ou de -1, ils évoluent de manière liée, mais inverse
        - ex : température extérieure et consommation de chauffage

Les fonctions OLAP de SQL

- Fonctions statistiques avec la 7.3
  - CORRELATION(x , y)

```
1  With temp
2    As (Select Year(logtime) As annee, Count(*) As nombre
3          From httplog.httplog
4          Where Year(logtime) < 2017
5          Group By Year(logtime), Month(logtime))
6    Select correlation(annee, nombre)
7    As cor_annee
8    From temp;
9
```

COR_ANNEE
0.70388607032265

Les fonctions OLAP de SQL

- Fonctions statistiques avec la 7.3
  - POURCENTILLE_CONT(xx) WITHIN GROUP(colonne)
    - retourne la valeur pour laquelle il y a xx lignes au dessus et (1-xx) lignes en dessous dans la sélection

```
11  With temp
12    As (Select Year(logtime) As annee, Count(*) As nombre
13          From httplog.httplog
14          where year(logtime) <2019
15          Group By Year(logtime))
16    Select percentile_cont(0.75) Within
17       Group (Order By nombre)
18       From temp;
```

00001
9896358.00

ANNEE	NOMBRE
2012	3903384
2013	6245919
2014	6815648
2015	7895074
2018	9630640
2017	10162076
2016	12080356

    - Valeur au ¾ (ayant 75 % des valeurs en dessous, 25 % au dessus)

Les fonctions OLAP de SQL

- Fonctions statistiques avec la 7.3
  - POURCENTILLE_DISC(xx) WITHIN GROUP(colonne)
    - retourne la valeur de la ligne pour laquelle il y a xx lignes au dessus et (1-xx) lignes en dessous dans la sélection

```
11  With temp
12    As (Select Year(logtime) As annee, Count(*) As nombre
13          From httplog.httplog
14          where year(logtime) <2019
15          Group By Year(logtime))
16    Select percentile_disc(0.75) Within
17       Group (Order By nombre) as trois_quart
18       From temp;
```

TROIS_QUART
9630640

ANNEE	NOMBRE
2012	3903384
2013	6245919
2014	6815648
2015	7895074
2018	9630640
2017	10162076
2016	12080356

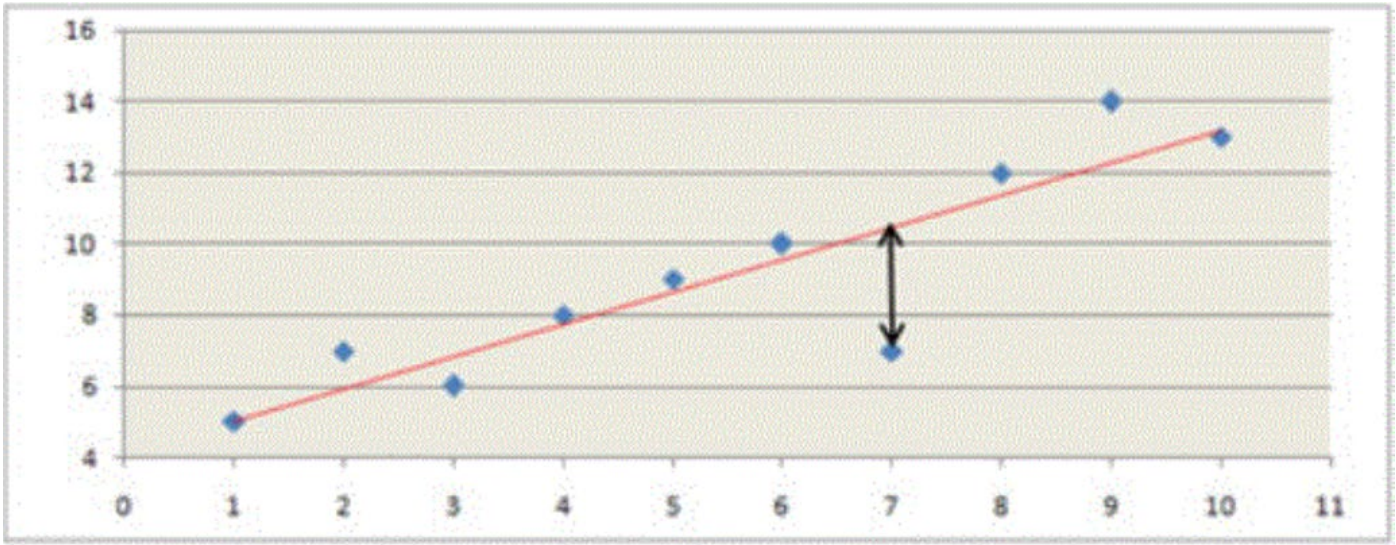
    - Ligne au ¾ (ayant 75 % des lignes en dessous, 25 % au dessus)

Les fonctions OLAP de SQL

- Fonctions statistiques avec la 7.3
  - Régression linéaire
    - Il s'agit de voir s'il y a une relation entre deux valeurs la température baisse-t-elle quand la l'altitude augmente, dans une région donnée ?(C'est probable)
    - Le taux d'émission de gaz à effet de serre d'un pays est-il lié à son PIB ?
    - Voyons si nous pouvons tracer une courbe(régression) ou une droite (régression linéaire) ayant
      - sur l'axe des X l'altitude ou le PIB
      - Sur l'axe des Y la température
      - ou le taux d'émission des G.E.S

Les fonctions OLAP de SQL

- Fonctions statistiques avec la 7.3
  - La droite de régression



Les fonctions OLAP de SQL

- Fonctions statistiques avec la 7.3
  - REGR_COUNT() -> Nbr de paires non nul
  - REGR_INTERCEPT /REGR_ICPT (ordonnée à l'origine (valeur de x quand y=0), ou constante de régression)
  - REGR_R2 (coefficient de détermination, proche de 1, les points sont tous près de la droite)
  - REGR_SLOPE (la pente)
  - REGR_AVGX (moyenne de x, après élimination des valeurs nulles)
  - REGR_AVGY (moyenne de y, après élimination des valeurs nulles)
  - REGR_SXX (REGR_COUNT(X, Y) * VARIANCE(Y))
  - REGR_SXY (REGR_COUNT(X, Y) * COVARIANCE(X, Y))
  - REGR_SYY (REGR_COUNT(X, Y) * VARIANCE(X))

Les fonctions OLAP de SQL

- Fonctions statistiques avec la 7.3
  - REGR_R2 (coefficient de détermination, proche de 1, les points sont tous près de la droite)

```
11  With temp
12    As (Select Year(logtime) As annee, Count(*) As nombre
13          From httplog.httplog
14          where year(logtime) <2019
15          Group By Year(logtime))
16    Select regr_r2(annee, nombre)
17       From temp;
```

00001
0.7247426555740303

Les fonctions OLAP de SQL

- Fonctions statistiques avec la 7.3
  - REGR_SLOPE (la pente)

```
11  With temp
12    As (Select Year(logtime) As annee, Count(*) As nombre
13          From httplog.httplog
14          where year(logtime) <2019
15          Group By Year(logtime))
16    Select dec(regr_slope(annee, nombre), 11, 9)
17       From temp;
```

00001
0.000000670

  - Ici faible, mais montante

Les fonctions OLAP de SQL

- Fonctions statistiques avec la 7.3
  - REGR_INTERCEPT (ordonnées à l'origine)
    - Année **théorique** où nous étions à 0

```
11  With temp
12    As (Select Year(logtime) As annee, Count(*) As nombre
13          From httplog.httplog
14          where year(logtime) <2019
15          Group By Year(logtime))
16    Select int(regr_intercept(annee, nombre)) as annee_zero
17       From temp;
```

ANNEE_ZERO
2009

ANNEE	NOMBRE
2012	3903384
2013	6245919
2014	6815648
2015	7895074
2016	12080356
2017	10162076
2018	9630640

Les fonctions OLAP de SQL

- Fonctions OLAP avec la 7.3

  - LAG()
  - LEAD()
  - NTILE()
  - CUME_DSIT()

  - PERCENT_RANK()

Les fonctions OLAP de SQL

- Fonctions OLAP avec la 7.3
  - LAG() et LEAD()
    - LAG() la valeur du dessus
    - LEAD() la valeur du dessous

```
11  With temp
12    As (Select Year(logtime) As annee, Count(*) As nombre
13          From httplog.httplog
14          where year(logtime) <2019
15          Group By Year(logtime))
16    Select annee, nombre, (nombre - lag(nombre) over (order by annee)) as evolution,
17    (nombre - lead(nombre) over (order by annee)) as inverse
18       From temp;
```

ANNEE	NOMBRE	EVOLUTION	INVERSE
2012	3903384	-	-2342535
2013	6245919	2342535	-569729
2014	6815648	569729	-1079426
2015	7895074	1079426	-4185282
2016	12080356	4185282	1918280
2017	10162076	-1918280	531436
2018	9630640	-531436	-

Les fonctions OLAP de SQL

- Fonctions OLAP avec la 7.3
  - NTILE(4) le quartille / NTILE(10) le décille, etc...

SOMME
56733097

```
11  With temp
12    As (Select Year(logtime) As annee, Count(*) As nombre
13          From httplog.httplog
14          where year(logtime) <2019
15          Group By Year(logtime))
16    Select annee, nombre, ntile(3) over (order by nombre desc) as quel_tiers,
17    ntile(4) over (order by nombre desc) as quel_quart
18      From temp;
```

ANNEE	NOMBRE	QUEL_TIERS	QUEL_QUART
2016	12080356	1	1
2017	10162076	1	1
2018	9630640	1	2
2015	7895074	2	2
2014	6815648	2	3
2013	6245919	3	3
2012	3903384	3	4

Les fonctions OLAP de SQL

- Fonctions OLAP avec la 7.3
  - CUME_DIST, distribution cumulée (le dernier valant 1 ici multiplié par 100 pour plus de lisibilité)

```
11  With temp
12    As (Select Year(logtime) As annee, Count(*) As nombre
13          From httplog.httplog
14          Where Year(logtime) < 2019
15          Group By Year(logtime))
16    Select annee, nombre, int(cume_dist()
17       over(Order By nombre) * 100)
18    As Dist_cumul From temp;
```

ANNEE	NOMBRE	DIST_CUMUL
2012	3903384	14
2013	6245919	28
2014	6815648	42
2015	7895074	57
2018	9630640	71
2017	10162076	85
2016	12080356	100

Les fonctions OLAP de SQL

- Fonctions OLAP avec la 7.3 TR1
  - PERCENT_RANK(), % du rang (le dernier valant 1, le premier 0)
    - Classement en pourcentage

```
11  With temp
12    As (Select Year(logtime) As annee, Count(*) As nombre
13          From httplog.httplog
14          Where Year(logtime) < 2019
15          Group By Year(logtime))
16    Select annee, nombre, int(percent_rank()
17       over(Order By nombre) * 100)
18    As rang_pourcent From temp;
```

ANNEE	NOMBRE	RANG_POURCENT
2012	3903384	0
2013	6245919	16
2014	6815648	33
2015	7895074	50
2018	9630640	66
2017	10162076	83
2016	12080356	100

Les fonctions OLAP de SQL

- Fonctions OLAP avec la 7.3
  - Fonctions d'agrégation
    - La première/dernière valeur de x
      - FIRST_VALUE(x) OVER (order by ...)
      - LAST_VALUE(X) OVER (order by …)
    - La nième valeur de x
      - NTH_VALUE(x , n)
    - % de la somme
      - RATIO_TO_REPORT(x)

Les fonctions OLAP de SQL

- Fonctions OLAP avec la 7.3
  - FIRST_VALUE(x) OVER (order by ...)

```
14  With temp
15    As (Select Year(logtime) As annee, Count(*) As nombre
16          From httplog.httplog
17          Where Year(logtime) < 2019
18          Group By Year(logtime))
19    Select annee, nombre, first_value(nombre) over(Order By annee)
20    As valeur_1 , nombre * 100 / first_value(nombre) over(Order By annee)
21    As fois_plus From temp;
```

ANNEE	NOMBRE	VALEUR_1	FOIS_PLUS
2012	3903384	3903384	100
2013	6245919	3903384	160
2014	6815648	3903384	174
2015	7895074	3903384	202
2016	12080356	3903384	309
2017	10162076	3903384	260
2018	9630640	3903384	246

Les fonctions OLAP de SQL

- Fonctions OLAP avec la 7.3
  - LAST_VALUE(x) OVER (order by ...)

```
14  With temp
15    As (Select Year(logtime) As annee, Count(*) As nombre
16          From httplog.httplog
17          Where Year(logtime) < 2019
18          Group By Year(logtime))
19    Select annee, nombre, last_value(nombre) over()
20    As valeur_finale , nombre * 100 / last_value(nombre) over()
21    As ratio From temp;
```

ANNEE	NOMBRE	VALEUR_FINALE	RATIO
2012	3903384	9630640	40
2013	6245919	9630640	64
2014	6815648	9630640	70
2015	7895074	9630640	81
2016	12080356	9630640	125
2017	10162076	9630640	105
2018	9630640	9630640	100

Les fonctions OLAP de SQL

- Fonctions OLAP avec la 7.3
  - RATIO_TO_REPORT(colonne) : % de la somme

SOMME
56733097

```
14  With temp
15    As (Select Year(logtime) As annee, Count(*) As nombre
16          From httplog.httplog
17          Where Year(logtime) < 2019
18          Group By Year(logtime))
19    Select annee, nombre, dec(ratio_to_report(nombre) over() * 100, 5, 2)
20    As  ratio From temp;
```

ANNEE	NOMBRE	RATIO
2012	3903384	6.88
2013	6245919	11.01
2014	6815648	12.01
2015	7895074	13.92
2016	12080356	21.29
2017	10162076	17.91
2018	9630640	16.98

Les fonctions OLAP de SQL

- Fonctions OLAP avec la 7.3
  - Toutes les fonctions d'agrégation ("historiques" ou OLAP)
    - Acceptent la clause OVER()

```
14  With temp
15    As (Select Year(logtime) As annee, Count(*) As nombre
16          From httplog.httplog
17          Where Year(logtime) < 2019
18          Group By Year(logtime))
19    Select annee, sum(nombre) over(order by annee) From temp;
```

ANNEE	00002
2012	3903384
2013	10149303
2014	16964951
2015	24860025
2016	36940381
2017	47102457
2018	56733097

Les fonctions OLAP de SQL

- Fonctions OLAP avec la 7.3
  - Toutes les fonctions d'agrégation ("historiques" ou OLAP)
    - Acceptent le partitionnement

```
14  With temp
15    As (Select Year(logtime) As annee, Month(logtime) As mois, Count(*) As nombre
16          From httplog.httplog
17          Where Year(logtime) < 2019
18          Group By Year(logtime), month(logtime))
19    Select annee, mois, Sum(nombre) over( partition by mois Order By annee)
20       From temp;
21
```

ANNEE	MOIS	00003
2012	1	356076
2013	1	795304
2014	1	1336374
2015	1	1998547
2016	1	2783261
2017	1	4398342
2018	1	5097231
2012	2	342810
2013	2	802901
2014	2	1327389
2015	2	1990386
2016	2	2822874

Les fonctions OLAP de SQL

- Fonctions OLAP avec la 7.3
  - Toutes les fonctions d'agrégation ("historiques" ou OLAP)
    - Acceptent le fenétrage
      - ROWS → fenêtre basée sur les x lignes précédentes et suivantes
      - RANGE → fenêtre basée sur les x valeurs de clé précédentes et suivantes
    - Vous pouvez indiquer :
      - Une position de départ (sous-entendu jusqu'à la ligne en cours)
      - Une position d'arrivée (sous-entendu à partir de la ligne en cours)
      - Une plage avec BETWEEN Début AND Fin

Les fonctions OLAP de SQL

- Fonctions OLAP avec la 7.3
  - Toutes les fonctions d'agrégation ("historiques" ou OLAP)
    - Acceptent le fenétrage
      - Début (position de départ)
        - UNBOUNDED PRECEDING
        - n PRECEDING (n lignes ou clés précédentes)
        - CURRENT ROW
      - Fin (position d'arrivée)
        - UNBOUNDED FOLLOWING
        - n FOLLOWING (n lignes ou clés suivantes)
        - CURRENT ROW

Les fonctions OLAP de SQL

- Fonctions OLAP avec la 7.3
  - Toutes les fonctions d'agrégation ("historiques" ou OLAP)
    - Acceptent le fenétrage

```
14  With temp
15    As (Select Year(logtime) As annee, Count(*) As nombre
16          From httplog.httplog
17          Where Year(logtime) < 2019
18          Group By Year(logtime))
19    Select annee, nombre, Sum(nombre) over(  Order By annee rows between 1 preceding and 1 following)
20       From temp;
21
```

ANNEE	NOMBRE	00003
2012	3903384	10149303
2013	6245919	16964951
2014	6815648	20956641
2015	7895074	26791078
2016	12080356	30137506
2017	10162076	31873072
2018	9630640	19792716

16964951 =
3903384 (2012)
+
6245919 (2013)
+
6815648 (2014)


MERCI