Dans ce compte rendu, vous trouverez les réponses aux exercices 1 et 2 du TP1. Chaque réponse est composée de la requête SQL, d'une capture d'écran du résultat et d'une explication de la requête.

Compte rendu du TP1

SQL OLAP

Mustapha NEZZARI

Sommaire

Exercice 1

Question a

Requête

```
SELECT deptno,
       ename,
       sal,
       Rank()
         OVER (
           partition BY deptno
           ORDER BY sal DESC) AS "RANG"
FROM   emp
WHERE  deptno = 10 OR deptno = 30;
```

Résultat

	DEPTNO	ENAME	SAL	RANG
1	10	KING	5000	1
2	10	CLARK	2450	2
3	10	MILLER	1300	3
4	30	BLAKE	2850	1
5	30	ALLEN	1600	2
6	30	TURNER	1500	3
7	30	MARTIN	1250	4
8	30	WARD	1250	4
9	30	JAMES	950	6

Explications

Cette requête classe les salaires des employés pour les départements 10 et 30. Ce classement est réalisé grâce à la fonction fenêtrée `Rank ()` qui utilise la clause `partition BY` pour faire des groupes de classements par département. La clause `ORDER BY sal` permet de définir sur quelle colonne le tri est fait dans chaque groupe, ici le salaire par ordre décroissant. On filtre enfin sur les départements 10 et 30 grâce à la clause `WHERE deptno = 10 OR deptno = 30`.

Question b

Requête

```
SELECT deptno,
       ename,
       sal,
       Dense_rank()
         OVER (
           partition BY deptno
           ORDER BY sal DESC) AS "RANG"
FROM   emp
WHERE  deptno = 10
      OR deptno = 30;
```

Résultat

	DEPTNO	ENAME	SAL	RANG
1	10	KING	5000	1
2	10	CLARK	2450	2
3	10	MILLER	1300	3
4	30	BLAKE	2850	1
5	30	ALLEN	1600	2
6	30	TURNER	1500	3
7	30	MARTIN	1250	4
8	30	WARD	1250	4
9	30	JAMES	950	5

Explications

Cette requête effectue la même chose que la requête précédente à la seule différence que les trous sont comblés dans le rang grâce à la fonction fenêtrée `Dense_rank ()`.

Question c

Requête

```
SELECT DISTINCT deptno,
                sal,
                Dense_rank()
                  over (
                    PARTITION BY deptno
                    ORDER BY sal DESC) AS "RANG"
FROM   emp
WHERE  deptno = 10
       OR deptno = 20
ORDER  BY deptno;
```

Résultat

	DEPTNO	SAL	RANG
1	10	5000	1
2	10	2450	2
3	10	1300	3
4	20	3000	1
5	20	2975	2
6	20	1100	3
7	20	800	4

Explications

Cette requête classe le salaire des employés par ordre décroissant pour les départements 10 et 20. On utilise la même requête qu'à la question précédente sauf que l'on filtre cette fois sur les départements 10 et 30 et que l'on a une clause `DISTINCT` pour ne pas avoir de doublons au niveau des salaires si par exemple deux employés gagnent le même salaire. Enfin la clause `ORDER BY deptno` permet d'ordonner le tout par département.

Question d

Requête

Avec le Group By

```
SELECT job,
       SUM(sal) AS "TOT_SAL_JOB"
FROM   emp
GROUP  BY job;
```

Avec le Partition by

```
SELECT DISTINCT job,
                SUM(sal)
                  over (
                    PARTITION BY job) AS "TOT_SAL_JOB"
FROM   emp;
```

Résultat

	JOB	TOT_SAL_JOB
1	CLERK	4150
2	SALESMAN	5600
3	PRESIDENT	5000
4	MANAGER	8275
5	ANALYST	6000

Explications

Cette requête permet de connaître le salaire total versé par profession. Il est possible de l'avoir par deux méthodes. La première en utilisant la fonction `SUM ()` sur les salaires et en faisant une agrégation sur les jobs. La deuxième en utilisant la fonction `SUM ()` et la clause `OVER` qui va nous permettre de définir une zone sur laquelle nous allons travailler. Cette zone est définie par la clause `PARTITION BY job` qui permet de réaliser des groupes. Ici on fait la somme des salaires par job, on veut donc un groupe par job pour pouvoir ensuite faire la somme des salaires dans chacun de ces groupes. Comme la clause `PARTITION BY` n'affecte pas le nombre de lignes retournées (cf question e), on a une ligne par entrée dans la table EMP. Il faut donc utiliser la clause DISTINCT pour ne pas avoir de doublon.

Question e

La différence entre la clause GROUP BY et PARTITION BY est que la clause GROUP BY permet de faire une agrégation selon certaines colonnes. Cela affecte donc le nombre de lignes retournées. La clause PARITION BY n'affecte pas le nombre de lignes retournées car elle n'agrège pas les lignes. Elle définit plutôt des groupes selon les colonnes sélectionnées. C'est pour cela qu'on l'utilise souvent avec la clause DISTINCT pour ne pas avoir de doublon.

Question f

Requête

```
SELECT deptno,
       job,
       SUM(sal)
FROM   emp
GROUP  BY rollup ( deptno, job );
```

Résultat

	DEPTNO	JOB	SUM(SAL)
1	10	CLERK	1300
2	10	MANAGER	2450
3	10	PRESIDENT	5000
4	10	(null)	8750
5	20	CLERK	1900
6	20	ANALYST	6000
7	20	MANAGER	2975
8	20	(null)	10875
9	30	CLERK	950
10	30	MANAGER	2850
11	30	SALESMAN	5600
12	30	(null)	9400
13	(null)	(null)	29025

Explications

Cette requête permet de connaître le montant total des salaires versés :

- Tout département et job confondu (en orange)
- Par département (en vert)
- Par département et job : (en bleu)

Ceci est possible grâce à la clause `GROUP BY rollup ( deptno, job )` qui va permettre à la fonction `SUM ()` de faire la somme sur des granularité de plus en plus grande, c'est-à-dire d'abord sur l'ensemble bleu, puis vert puis orange.

Question g

Requête

Avec NVL ()

```
SELECT Nvl(To_char(deptno), 'TousDep')   AS DEPARTEMENT,
       Nvl(To_char(job), 'TousEmployes') AS JOB,
       SUM(sal)
FROM   emp
GROUP  BY rollup ( deptno, job )
ORDER  BY deptno,
          SUM(sal) DESC;
```

Avec DECODE ()

```
SELECT Decode(deptno, NULL, 'TousDep', deptno) AS DEPARTEMENT,
       Decode(job, NULL, 'TousEmployes', job)  AS JOB,
       SUM(sal)
FROM   emp
GROUP  BY rollup ( deptno, job )
ORDER  BY deptno,
          SUM(sal) DESC;
```

Résultat

	DEPARTEMENT	JOB	SUM(SAL)
1	10	TousEmployes	8750
2	10	PRESIDENT	5000
3	10	MANAGER	2450
4	10	CLERK	1300
5	20	TousEmployes	10875
6	20	ANALYST	6000
7	20	MANAGER	2975
8	20	CLERK	1900
9	30	TousEmployes	9400
10	30	SALESMAN	5600
11	30	MANAGER	2850
12	30	CLERK	950
13	TousDep	TousEmployes	29025

Explications

Les deux requêtes font exactement la même chose qu'à la question précédente si ce n'est que les valeurs `(null)` sont remplacées par du texte grâce aux fonctions `NVL ()` et `DECODE ()`.

La fonction `NVL (texte, defaut)` prend en argument :

- `texte` : Le texte à afficher
- `defaut` : La valeur par défaut à afficher si le premier argument est `(null)`

La fonction `DECODE (expression, [recherche, resultat]+, defaut)` prend en argument :

- `expression` : L'expression dans laquelle on va rechercher des valeurs
- `[recherche, resultat]` : Un bloc qui peut se répéter, qui comprend deux arguments :
    - `recherche` : La valeur à rechercher (et à remplacer par l'argument suivant) dans l'expression
    - `resultat` : La valeur qui remplacera l'argument précédent s'il est trouvé
- `defaut` : La valeur par défaut si aucune des expressions du bloc précédent n'est trouvée.

Exercice 2

Question 1

Requête

```
SELECT annee,
       cl_r,
       category,
       Avg(qte * pu) AS CA_MOYEN
FROM   ventes
       join clients
         ON clients.cl_id = ventes.cid
       join produits
         ON ventes.pid = produits.pid
       join temps
         ON ventes.tid = temps.tid
WHERE  annee = 2009
    OR annee = 2010
GROUP  BY rollup ( annee, cl_r, category );
```

Résultat

	ANNEE	CL_R	CATEGORY	CA_MOYEN
654	2010	05454-876	Condiments	3789
655	2010	05454-876	Pâtes et céréales	1100
656	2010	05454-876	Poissons et fruits de mer	3800
657	2010	05454-876	(null)	3665,1875
658	2010	Isle of Wight	Boissons	1500
659	2010	Isle of Wight	Desserts	361,5
660	2010	Isle of Wight	Pâtes et céréales	1170
661	2010	Isle of Wight	Poissons et fruits de mer	2850
662	2010	Isle of Wight	(null)	1226,14285...
663	2010	Nueva Esparta	Viandes	1987,5
664	2010	Nueva Esparta	Boissons	2536
665	2010	Nueva Esparta	Desserts	1690,66666...
666	2010	Nueva Esparta	Condiments	1670,08333...
667	2010	Nueva Esparta	Produits laitiers	2880
668	2010	Nueva Esparta	Pâtes et céréales	10440
669	2010	Nueva Esparta	(null)	2719,26785...
670	2010	(null)	(null)	3257,22376...
671	(null)	(null)	(null)	3158,31296...

Explications

Cette requête permet de connaître la moyenne des ventes par :

- année, région et catégorie (bleu)
- année et région (vert)
- année (orange)
- Toute année région et catégorie confondue (jaune)

pour les années 2009 et 2010. Pour ça on utilise la clause `GROUP BY rollup ( annee, cl_r, category )`. Le filtre sur les années 2009 et 2010 est réalisé grâce à la clause `WHERE annee = 2009 OR annee = 2010`. Enfin pour pouvoir utiliser les informations des autres tables, on utilise des jointures sur les id.

Question 2

Requête

```
SELECT annee,
       cl_r,
       category,
       Avg(qte * pu) AS CA_MOYEN
FROM   ventes
       join clients
         ON clients.cl_id = ventes.cid
       join produits
         ON ventes.pid = produits.pid
       join temps
         ON ventes.tid = temps.tid
WHERE  annee = 2009
        OR annee = 2010
GROUP  BY cube ( annee, cl_r, category );
```

Résultat

	ANNEE	CL_R	CATEGORY	CA_MOYEN
1133	(null)	WX3 6FW	Boissons	2520
1134	(null)	WX3 6FW	Condiments	2395
1135	(null)	WX3 6FW	Desserts	2400
1136	(null)	WX3 6FW	Pâtes et céréales	2422,5
1137	(null)	WX3 6FW	Viandes	1941,75
1138	(null)	WX3 6FW	(null)	2356,6785714...
1139	(null)	WY	Boissons	21080
1140	(null)	WY	Desserts	380
1141	(null)	WY	Poissons et fruits de mer	2280
1142	(null)	WY	Produits laitiers	944
1143	(null)	WY	Produits secs	1200
1144	(null)	WY	Pâtes et céréales	2186,6666666...
1145	(null)	WY	Viandes	3096,8333333...
1146	(null)	WY	(null)	3047,9
1147	(null)	(null)	Boissons	4763,0748299...
1148	(null)	(null)	Condiments	3071,7767857...
1149	(null)	(null)	Desserts	2487,1837837...
1150	(null)	(null)	Poissons et fruits de mer	2816,1734693...
1151	(null)	(null)	Produits laitiers	3330,5127551...
1152	(null)	(null)	Produits secs	3871,1228070...
1153	(null)	(null)	Pâtes et céréales	3330,0420792...
1154	(null)	(null)	Viandes	2250,3220338...
1155	(null)	(null)	(null)	3158,3129646...

	ANNEE	CL_R	CATEGORY	CA_MOYEN
665	2010	WX1 6LT	Viandes	24
666	2010	WX1 6LT	(null)	724,66666666...
667	2010	WX3 6FW	Boissons	2520
668	2010	WX3 6FW	Condiments	2395
669	2010	WX3 6FW	Desserts	2400
670	2010	WX3 6FW	Pâtes et céréales	525
671	2010	WX3 6FW	Viandes	1941,75
672	2010	WX3 6FW	(null)	2029,4583333...
673	2010	WY	Desserts	375
674	2010	WY	Produits laitiers	1180
675	2010	WY	Pâtes et céréales	3600
676	2010	WY	Viandes	4231,25
677	2010	WY	(null)	2723,5
678	2010	(null)	Boissons	4529,3465346...
679	2010	(null)	Condiments	3065,2865853...
680	2010	(null)	Desserts	2588,8703703...
681	2010	(null)	Poissons et fruits de mer	3141,88
682	2010	(null)	Produits laitiers	3588,9178571...
683	2010	(null)	Produits secs	4278,7380952...
684	2010	(null)	Pâtes et céréales	3542,3007246...
685	2010	(null)	Viandes	2332,1860465...
686	2010	(null)	(null)	3257,2237694...
687	(null)	01-012	Boissons	813,33333333...

Explications

Cette requête permet d'avoir la moyenne des ventes par année, selon les dimensions région et catégorie pour les années 2009 et 2010. Il a donc fallu utiliser un GROUP BY CUBE afin d'obtenir tous ces sous totaux. Cette clause ressemble à un rollup sur toutes les combinaisons possibles entre l'année, la région et la catégorie. On obtient un résultat de ce type :

- Année, région, catégorie (gris)
- Année, région (violet)
- Année, catégorie (cyan)
- Année (marron beige)
- Région, catégorie (bleu)
- Région (vert)
- Catégorie (orange)

  Toute année, région et catégorie confondu (jaune)

Question 3

Requête

```
SELECT annee,
       category,
       pname
FROM   (SELECT t.annee,
               p.category,
               p.pname,
               Rank()
                 over (
                   PARTITION BY t.annee, p.category
                   ORDER BY SUM(qte*pu) DESC) RANG
        FROM   ventes v
               join produits p
                 ON v.pid = p.pid
               join temps t
                 ON v.tid = t.tid
        GROUP  BY t.annee,
                  p.category,
                  p.pname)
WHERE  rang = 1;
```

Résultat

	ANNEE	CATEGORY	PNAME
1	2009	Boissons	Côte de Blaye
2	2009	Condiments	Northwoods Cranberry Sauce
3	2009	Desserts	Tarte au sucre
4	2009	Pâtes et céréales	Raclette Courdavault
5	2009	Poissons et fruits de mer	Wimmers gute Semmelknodel
6	2009	Produits laitiers	Alice Mutton
7	2009	Produits secs	Manjimup Dried Apples
8	2009	Viandes	Carnarvon Tigers
9	2010	Boissons	Côte de Blaye
10	2010	Condiments	Sirop d érable
11	2010	Desserts	Tarte au sucre
12	2010	Pâtes et céréales	Raclette Courdavault
13	2010	Poissons et fruits de mer	Gnocchi di nonna Alice
14	2010	Produits laitiers	Alice Mutton
15	2010	Produits secs	Manjimup Dried Apples
16	2010	Viandes	Carnarvon Tigers
17	2011	Boissons	Côte de Blaye
18	2011	Condiments	Vegie-spread
19	2011	Desserts	Tarte au sucre
20	2011	Pâtes et céréales	Raclette Courdavault
21	2011	Poissons et fruits de mer	Wimmers gute Semmelknodel
22	2011	Produits laitiers	Alice Mutton
23	2011	Produits secs	Uncle Bob s Organic Dried Pears
24	2011	Viandes	Carnarvon Tigers

Explications

Cette requête permet de connaître pour chaque année le produit le mieux vendu par catégorie. Pour cela, on fait dans un premier temps la somme des ventes par produit, par catégorie et pour chaque année et dans un second temps on prend le produit où le chiffre d'affaire est le plus grand par catégorie et par année.

La première étape est réalisée grâce à une sous requête qui classe les produits par catégorie et par année selon leur chiffre d'affaire. Ce classement est fait par la fonction `Rank() over ( PARTITION BY t.annee, p.category ORDER BY SUM(qte*pu) DESC)` RANG. La clause PARITION BY permet de faire des groupe d'année et dans chacun de ces groupes, des groupes de catégorie. On peut ensuite faire le classement des produits, dans chaque groupe de catégorie, par leur chiffre d'affaire grâce à la clause `ORDER BY SUM(qte*pu) DESC`.

La deuxième étape consiste à ne prendre que les premières lignes de chaque classement qui correspondent aux produits ou le chiffre d'affaire est le plus grand par année et par catégorie.

Question 4

Requête

```
SELECT t.annee,
       p.category,
       SUM(v.qte * v.pu) AS CA_TOTAL
FROM   ventes v
       join produits p
         ON v.pid = p.pid
       join temps t
         ON v.tid = t.tid
GROUP  BY rollup ( t.annee, p.category )
HAVING Grouping_id(t.annee, p.category) != 3
ORDER  BY t.annee,
          ca_total DESC;
```

Résultat

	ANNEE	CATEGORY	CA_TOTAL
1	2009	(null)	883768
2	2009	Boissons	242708
3	2009	Pâtes et céréales	183831
4	2009	Desserts	110631,5
5	2009	Viandes	97455
6	2009	Condiments	92685,5
7	2009	Produits laitiers	75166
8	2009	Produits secs	40947
9	2009	Poissons et fruits de mer	40344
10	2010	(null)	2514576,75
11	2010	Pâtes et céréales	488837,5
12	2010	Boissons	457464
13	2010	Desserts	349497,5
14	2010	Viandes	300852
15	2010	Condiments	251353,5
16	2010	Produits laitiers	251224,25
17	2010	Poissons et fruits de mer	235641
18	2010	Produits secs	179707
19	2011	(null)	1688292,7
20	2011	Boissons	480950
21	2011	Pâtes et céréales	342882
22	2011	Desserts	228732,5
23	2011	Viandes	200525,2
24	2011	Condiments	131237,25
25	2011	Poissons et fruits de mer	118536,25
26	2011	Produits secs	94246,25
27	2011	Produits laitiers	91183,25

Explications

Cette requête permet de connaitre le total du CA par :

- année et catégorie (bleu)
- année (vert)

Cela a comme jusqu'à présent été possible grâce à la clause ROLLUP. Cependant pour enlever le sous-total « toute année et catégorie confondue » il a fallu utiliser les GROUPING_ID qui permettent de connaître le niveau d'agrégation pour chaque ligne de sous-total. Ici on peut représenter notre cas avec ce tableau :

Niveau d'agrégation	Bit vector	GROUPING_ID
Année, catégorie	0 0	0
Année	0 1	1
Catégorie	1 0	2
Grand total	1 1	3

Ici nous voulons cacher le Grand total, il faut donc utiliser une clause qui exclut le GROUPING_IG n° 3 comme la clause `HAVING Grouping_id(t.annee, p.category) != 3`.

Question 5

Requête

```
SELECT annee,
       mois,
       ca_total
FROM   (SELECT t.annee,
               t.mois,
               p.pname,
               SUM(qte * pu)                  AS CA_TOTAL,
               Rank()
                 over (
                   PARTITION BY t.annee
                   ORDER BY SUM(qte*pu) DESC) AS RANG
        FROM   ventes v
               join produits p
                 ON v.pid = p.pid
               join temps t
                 ON v.tid = t.tid
        WHERE  p.pname LIKE 'Sirop d érable'
        GROUP  BY t.annee,
                  t.mois,
                  p.pname
        ORDER  BY t.annee,
                  t.mois)
WHERE  rang = 1;
```

Résultat

	ANNEE	MOIS	CA_TOTAL
1	2010	7	17100
2	2011	4	13822,5

Explications

Cette requête permet de connaître le meilleur mois de vente du produit « Sirop d'érable » pour chaque année. Pour cela, on fait dans un premier temps la somme des ventes par mois pour chaque année et dans un second temps on prend le mois où le chiffre d'affaire est le plus grand par année.

La première étape est réalisée grâce à une sous requête qui fait la somme par mois et par année et pour le produit Sirop d'érable (`WHERE  p.pname LIKE 'Sirop d érable'`). On ajoute ensuite une colonne qui permet de voir le classement des mois. Ceci est fait par la fonction RANK qui donne le rang des mois par année grâce à la clause PARTITION BY. Ce rang est trié par le chiffre d'affaire pour placer le mois qui a le plus grand chiffre d'affaire en tête, grâce à la clause `ORDER BY SUM(qte*pu) DESC`.

La deuxième étape consiste à ne prendre que les premières lignes de chaque classement qui correspondent aux mois ou le chiffre d'affaire est le plus grand par année pour le produit demandé.

Question 6

Requête

```
SELECT t.annee,
       c.cl_name,
       p.category,
       SUM(qte * pu) AS CA_TOTAL
FROM   ventes v
       join clients c
         ON c.cl_id = v.cid
       join produits p
         ON v.pid = p.pid
       join temps t
         ON v.tid = t.tid
GROUP  BY grouping sets ( ( t.annee, c.cl_name ), ( t.annee, p.category ) );
```

Résultat

	ANNEE	CL_NAME	CATEGORY	CA_TOTAL
192	2011	Que Delicia	(null)	6768
193	2011	LILA-Supermercado	(null)	28995,3
194	2010	Hanari Carnes	(null)	29321,5
195	2011	Hungry Owl All-Night Gr...	(null)	27328,7
196	2009	Que Delicia	(null)	9800
197	2010	Vins et alcools Chevalier	(null)	1899
198	2009	Seven Seas Imports	(null)	25512
199	2010	Old World Delicatessen	(null)	29480
200	2009	Old World Delicatessen	(null)	13082,5
201	2009	Bon app	(null)	21012,5
202	2010	Franchi S.p.A.	(null)	809,5
203	2010	Maison Dewey	(null)	23860
204	2011	Cactus Comidas para llevar	(null)	4535
205	2010	Laughing Bacchus Wine C...	(null)	945
206	2011	(null)	Boissons	480950
207	2010	(null)	Condiments	251353,5
208	2010	(null)	Desserts	349497,5
209	2009	(null)	Poisson...	40344
210	2011	(null)	Poisson...	118536,25
211	2010	(null)	Boissons	457464
212	2011	(null)	Produit...	94246,25
213	2010	(null)	Produit...	251224,25
214	2010	(null)	Poisson...	235641
215	2011	(null)	Condiments	131237,25
216	2011	(null)	Desserts	228732,5
217	2009	(null)	Condiments	92685,5
218	2009	(null)	Produit...	40947
219	2009	(null)	Viandes	97455

Explications

Cette requête permet de donner tous les totaux de ventes par année et nom du client puis par année et catégorie. Ceci est possible grâce à la clause `GROUP BY grouping sets ( ( t.annee, c.cl_name ), ( t.annee, p.category ) )`.

On Remarque donc que l'on a utilisé la clause GROUPING SETS qui permet de préciser les sous-totaux que l'on veut calculer contrairement à la clause ROLLUP qui calcule un jeu prédéfini de sous-totaux.

Question 7

Requête

```
SELECT p.category,
       SUM(qte)                    AS QTE_VENDU_2010,
       Ntile(3)
         over (
           ORDER BY SUM(qte) DESC) AS TIERS
FROM   ventes v
       join produits p
         ON v.pid = p.pid
       join temps t
         ON v.tid = t.tid
WHERE  t.annee = 2010
GROUP  BY p.category;
```

Résultat

	CATEGORY	QTE_VENDU_2010	TIERS
1	Pâtes et céréales	3431	1
2	Desserts	3355	1
3	Viandes	2751	1
4	Boissons	2256	2
5	Condiments	2233	2
6	Poissons et fruits de mer	1878	2
7	Produits laitiers	1820	3
8	Produits secs	1046	3

Explications

Cette requête donne la répartition par tiers des catégories selon leurs quantités totales vendues en 2010. Pour cela, on filtre les quantité vendues en 2010, on en fait la somme par catégorie puis on répartie ces sommes dans 3 groupes grâce à la fonction `Ntile(3) over ( ORDER BY SUM(qte) DESC) AS TIERS`.

Question 8

Requête

```
SELECT category,
       mois,
       Min(tid)       AS JOUR1,
       Max(tid)       AS JOUR5,
       SUM(qte_jours) AS QTE_5_JOURS
FROM   (SELECT p.category,
               SUM(qte)             AS QTE_JOURS,
               t.mois,
               t.jour,
               t.tid,
               Dense_rank()
                 over (
                   PARTITION BY p.category, t.mois
                   ORDER BY t.jour) AS RANG
        FROM   ventes v
               join produits p
                 ON v.pid = p.pid
               join temps t
                 ON v.tid = t.tid
        WHERE  t.annee = 2010
        GROUP  BY t.mois,
                  p.category,
                  t.jour,
                  t.tid)
WHERE  rang BETWEEN 1 AND 5
GROUP  BY category,
          mois
HAVING Count(rang) = 5
ORDER  BY category,
          mois;
```

Résultat

	CATEGORY	MOIS	JOUR1	JOUR5	QTE_5_JOURS
1	Boissons	1	153	171	179
2	Boissons	2	188	205	94
3	Boissons	3	218	238	143
4	Boissons	4	255	273	51
5	Boissons	5	276	294	134
6	Boissons	7	339	361	77
7	Boissons	8	375	384	125
8	Boissons	9	405	420	103
9	Boissons	10	441	449	216
10	Boissons	11	482	505	138
11	Boissons	12	513	530	205
12	Condiments	1	153	171	191
13	Condiments	2	194	204	313
14	Condiments	4	246	266	196
15	Condiments	5	280	303	240

Explications

Cette requête permet de connaître la quantité de produits vendus pour chaque catégorie, les 5 premiers jours de chaque mois de 2010. J'ai choisi de prendre les 5 premiers jours de ventes et de ne prendre que les mois où j'ai vendu au moins pendant 5 jours. Enfin on connaît aussi la quantité vendu le premier jour (JOUR1) et le 5ème jour (JOUR5) ce qui n'aurait pas été possible si l'on n'avait que 3 jours de ventes par exemple (JOUR5 aurait été a null).

Dans un premier temps, on fait la somme des quantités vendues par jour et on affiche le rang de ces jours par catégorie et par mois grâce à la fonction `Dense_rank() over ( PARTITION BY p.category, t.mois ORDER BY t.jour) AS RANG`.
Dans un second temps on filtre :

- Pour n'avoir que l'année 2010
- Pour n'avoir que les mois où l'on a au moins 5 jours de vente

Puis on fait la somme des quantités des 5 premiers jours de vente grâce à la fonction `SUM(qte_jours) AS QTE_5_JOURS` et la clause `WHERE rang BETWEEN 1 AND 5 GROUP BY category, mois`.