

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.....	4
Question a	4
Requête	4
Résultat	4
Explications.....	4
Question b	5
Requête	5
Résultat	5
Explications.....	5
Question c	6
Requête	6
Résultat	6
Explications.....	6
Question d	7
Requête	7
Résultat	7
Explications.....	7
Question e	8
Question f.....	8
Requête	8
Résultat	8
Explications.....	8
Question g	9
Requête	9
Résultat	9
Explications.....	9
Exercice 2.....	11
Question 1	11
Requête	11
Résultat	11
Explications.....	11
Question 2	12
Requête	12
Résultat	12
Explications.....	12

Question 3	13
Requête	13
Résultat	13
Explications.....	14
Question 4	15
Requête	15
Résultat	15
Explications.....	16
Question 5	17
Requête	17
Résultat	17
Explications.....	17
Question 6	18
Requête	18
Résultat	18
Explications.....	19
Question 7	20
Requête	20
Résultat	20
Explications.....	20
Question 8	21
Requête	21
Résultat	21
Explications.....	22

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 20 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;
```

Avec une sous requête

```
SELECT DISTINCT job,  
               (SELECT SUM(sal)  
                FROM   emp E2  
                WHERE  E1.job = E2.job) AS "TOT_SAL_JOB"  
FROM   emp E1;
```

Résultat

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

Explications

A COMPLETER

Question e

A COMPLETER

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 petite, c'est-à-dire d'abord sur l'ensemble orange, puis vert puis bleu.

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, défaut) prend en argument :

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

La fonction DECODE (expression, [recherche, resultat]+, default) 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 :
 - o **recherche** : La valeur à rechercher (et à remplacer par l'argument suivant) dans l'expression
 - o **resultat** : La valeur qui remplacera l'argument précédent s'il est trouvé
- **default** : 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

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
422	(null)	OX15 4NB	Desserts	3455,66666...
423	(null)	OX15 4NB	Produits secs	3233
424	(null)	OX15 4NB	Produits laitiers	3458,58333...
425	(null)	OX15 4NB	Pâtes et céréales	5867,5
426	(null)	OX15 4NB	Poissons et fruits de mer	560
427	(null)	S-958 22	(null)	2899,75892...
428	(null)	S-958 22	Viandes	1568,4
429	(null)	S-958 22	Boissons	6586,75
430	(null)	S-958 22	Desserts	5059,5
431	(null)	S-958 22	Condiments	1321,33333...
432	(null)	S-958 22	Produits secs	1860
433	(null)	S-958 22	Produits laitiers	1112,125
434	(null)	S-958 22	Pâtes et céréales	2585
435	(null)	S-958 22	Poissons et fruits de mer	875
436	(null)	05432-043	(null)	1708
437	(null)	05432-043	Viandes	240
438	(null)	05432-043	Desserts	1200

Explications

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

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

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

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

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

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	🔍 JOUR 1	🔍 JOUR 5	🔍 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