Rapport
TP 7

Maîtrise de l'interrogation analytique des données (SQL)

Réalisé en binôme par :
BENSALAH Kawthar / ABBACI Khaled
Numero du binôme : 22
Master 2 IL - Groupe 1
USTHB 2019/2020

TP 7

Objectif du TP

Maîtrise de l'interrogation analytique des données (SQL)

Introduction

À travers ce TP, nous allons maîtriser l'interrogation analytique des données à travers des fonctions qui permettent d'extraire des informations de manière plus simple et performante.

Réponse 1

- **Activation de l'option timing de oracle :**

```
SQL> set timing on;
SQL>
```

- **Montants versés annuels par Wilaya, pour chaque type de compte :**

```
SQL> Select t.Année, d.CodeWilaya, f.CodeTypeCompte,
  2         sum(f.MontantV) as MontV
  3  from FOperation f, DAgence d, DTemps t
  4  where f.NumAgence = d.NumAgence
  5        and f.CodeTemps = t.CodeTemps
  6  group by t.Année, d.CodeWilaya, f.CodeTypeCompte
  7  order by t.Année, d.CodeWilaya, f.CodeTypeCompte;

ANNÉE           CODEWILAYA CODETYPECOMPTE      MONTV
--------------- ---------- -------------- ----------
2015                     1              1 37915390,5
2015                     1              2 36217469,3
2015                     2              1 33182835,6
2015                     2              2 30944139,5
2015                     3              1 29209853,9
2015                     3              2 26907072,9
2015                     4              1   26255215
2015                     4              2   23296202
2015                     5              1 54956439,7
2015                     5              2 48537878,6
2015                     6              1 23426047,6
```

- **Temps d'exécution :**

```
384 ligne(s) sélectionnée(s).

Ecoulé : 00 :00 :03.58
SQL>
```

Réponse 2

- **Introduction des sous totaux sur R1 avec la clause rollup by :**

```
SQL> Select t.Année, d.CodeWilaya, f.CodeTypeCompte,
  2         sum(f.MontantV) as MontV
  3  from FOperation f, DAgence d, DTemps t
  4  where f.NumAgence = d.NumAgence
  5        and f.CodeTemps = t.CodeTemps
  6  group by rollup (t.Année, d.CodeWilaya, f.CodeTypeCompte)
  7  order by t.Année, d.CodeWilaya, f.CodeTypeCompte;

ANNÉE           CODEWILAYA CODETYPECOMPTE      MONTV
--------------- ---------- -------------- ----------
2015                     1              1 37915390,5
2015                     1              2 36217469,3
2015                     1                74132859,9
2015                     2              1 33182835,6
2015                     2              2 30944139,5
2015                     2                64126975,2
2015                     3              1 29209853,9
2015                     3              2 26907072,9
2015                     3                56116926,8
2015                     4              1   26255215
2015                     4              2   23296202
.
.
.
2018                    47              1 26085280,9
2018                    47              2 23236311,6
2018                    47                49321592,4
2018                    48              1 72198274,7
2018                    48              2 60736706,1
2018                    48                 132934981
2018                                      4230932776
                                          1,6887E+10
```

Remarque 1 : ROLLUP permet à l'instruction SELECT (Requête 1) de calculer plusieurs niveaux de sous-totaux parmi un ensemble de dimensions. Dans cet exemple, ROLLUP est spécifié sur les colonnes de regroupement (Année, CodeWilaya et CodeTypeCompte) et donc ROLLUP a créé des sous-totaux de 3+1 niveaux.

- **Temps d'exécution :**

```
581 ligne(s) sélectionnée(s).

Ecoulé : 00 :00 :05.59
SQL>
```

Réponse 3

- **Introduction des sous totaux sur R1 avec la clause cube by :**

```
SQL> Select t.Année, d.CodeWilaya, f.CodeTypeCompte,
  2          sum(f.MontantV) as MontV
  3  from FOperation f, DAgence d, DTemps t
  4  where f.NumAgence = d.NumAgence
  5        and f.CodeTemps = t.CodeTemps
  6  group by cube (t.Année, d.CodeWilaya, f.CodeTypeCompte)
  7  order by t.Année, d.CodeWilaya, f.CodeTypeCompte;

ANNÉE           CODEWILAYA CODETYPECOMPTE      MONTV
--------------- ---------- -------------- ----------
2015                     1              1 37915390,5
2015                     1              2 36217469,3
2015                     1                74132859,9
2015                     2              1 33182835,6
2015                     2              2 30944139,5
2015                     2                64126975,2
2015                     3              1 29209853,9
2015                     3              2 26907072,9
2015                     3                56116926,8
2015                     4              1   26255215
2015                     4              2   23286203
.
.
.
                        47              1  101942891
                        47              2 92926139,7
                        47                194869031
                        48              1  289072076
                        48              2  246092988
                        48                535165064
                                        1 8897457388
                                        2 7989767636
                                          1,6887E+10
```

Remarque 2 : CUBE dans cette requête a pris l'ensemble de colonnes de regroupement (Année, CodeWilaya et CodeTypeCompte) et a créé des sous-totaux pour chacune des combinaisons possibles. l'ensemble des résultats retournés par CUBE est le même que celui de ROLLUP plus toutes les combinaisons supplémentaires.

- **Temps d'exécution :**

```
735 ligne(s) sélectionnée(s).

Ecoulé : 00 :00 :06.25
SQL>
```

Réponse 4

- Introduction de la fonction grouping pour chaque dimension dans R2 :

```
SQL> Select t.Année, d.CodeWilaya, f.CodeTypeCompte,
  2         sum(f.MontantV) as MontV,
  3  grouping (t.Année) as An, grouping (d.CodeWilaya) as Cw,
  4         grouping (f.CodeTypeCompte) as Tc
  5  from FOperation f, DAgence d, DTemps t
  6  where f.NumAgence = d.NumAgence
  7         and f.CodeTemps = t.CodeTemps
  8  group by rollup (t.Année, d.CodeWilaya, f.CodeTypeCompte)
  9  order by t.Année, d.CodeWilaya, f.CodeTypeCompte;

ANNÉE           CODEWILAYA CODETYPECOMPTE      MONTV         AN         CW         TC
--------------- ---------- -------------- ---------- ---------- ---------- ----------
2015                     1              1 37915390,5          0          0          0
2015                     1              2 36217469,3          0          0          0
2015                     1                74132859,9          0          0          1
2015                     2              1 33182835,6          0          0          0
2015                     2              2 30944139,5          0          0          0
2015                     2                64126975,2          0          0          1
2015                     3              1 29209853,9          0          0          0
2015                     3              2 26907072,9          0          0          0
2015                     3                56116926,8          0          0          1
2015                     4              1   26255215          0          0          0
```

.

.

.

```
2018                    47              1 26085280,9          0          0          0
2018                    47              2 23236311,6          0          0          0
2018                    47                49321592,4          0          0          1
2018                    48              1 72198274,7          0          0          0
2018                    48              2 60736706,1          0          0          0
2018                    48                 132934981          0          0          1
2018                                      4230932776          0          1          1
                                          1,6887E+10          1          1          1
```

Remarque 3 : Dans cette requête, nous avons utilisé la fonction grouping qui retourne 1 quand elle rencontre NULL, créé par ROLLUP indiquant un sous-total et elle retourne 0 en cas de rencontre d'une autre valeur incluant une valeur stockée NULL.

- Temps d'exécution :

```
581 ligne(s) sélectionnée(s).

Ecoulé : 00 :00 :05.74
SQL>
```

Réponse 5

- Remplacement de la fonction grouping par la fonction grouping_id :

```
SQL> Select t.Année, d.CodeWilaya, f.CodeTypeCompte,
  2         sum(f.MontantV) as MontV,
  3  grouping_id (t.Année, d.CodeWilaya, f.CodeTypeCompte) as GID
  4  from FOperation f, DAgence d, DTemps t
  5  where f.NumAgence = d.NumAgence
  6         and f.CodeTemps = t.CodeTemps
  7  group by rollup (t.Année, d.CodeWilaya, f.CodeTypeCompte)
  8  order by t.Année, d.CodeWilaya, f.CodeTypeCompte;

ANNÉE           CODEWILAYA CODETYPECOMPTE      MONTV        GID
--------------- ---------- -------------- ---------- ----------
2015                     1              1 37915390,5          0
2015                     1              2 36217469,3          0
2015                     1                74132859,9          1
2015                     2              1 33182835,6          0
2015                     2              2 30944139,5          0
2015                     2                64126975,2          1
2015                     3              1 29209853,9          0
2015                     3              2 26907072,9          0
2015                     3                56116926,8          1
2015                     4              1   26255215          0
```

.
.
.

```
2018                    47              1 26085280,9          0
2018                    47              2 23236311,6          0
2018                    47                49321592,4          1
2018                    48              1 72198274,7          0
2018                    48              2 60736706,1          0
2018                    48                 132934981          1
2018                                      4230932776          3
                                          1,6887E+10          7
```

Remarque 4 : Le fait de remplacer la fonction GROUPING par la fonction GROUPING_ID a permis un gain d'espace de stockage car cette fonction retourne une valeur unique qui permet de déterminer le niveau exact du GROUP BY. Dans cet exemple, les valeurs possible varient de 0 à 7.

- Temps d'exécution :

```
581 ligne(s) sélectionnée(s).

Ecoulé : 00 :00 :05.71
SQL>
```

Réponse 6

- Amélioration de la lisibilité de la requête en utilisant la fonction decode :

```
SQL> Select decode(grouping (t.Année),1,'Total_A',t.Année)as an,
  2  decode(grouping (d.CodeWilaya),1,'Total_W',d.CodeWilaya)as cdw,
  3  decode(grouping (f.CodeTypeCompte),1,'Total_T',f.CodeTypeCompte) as cdt
  4  from FOperation f, DAgence d, DTemps t
  5  where f.NumAgence = d.NumAgence
  6          and f.CodeTemps = t.CodeTemps
  7  group by rollup(t.Année, d.CodeWilaya, f.CodeTypeCompte)
  8  order by t.Année, d.CodeWilaya, f.CodeTypeCompte;

AN              CDW                                    CDT
--------------- -------------------------------------- --------------------
2015            1                                      1
2015            1                                      2
2015            1                                      Total_T
2015            2                                      1
2015            2                                      2
2015            2                                      Total_T
2015            3                                      1
2015            3                                      2
2015            3                                      Total_T
2015            4                                      1
2015            4                                      2
```

.
.
.

```
2018            47                                     1
2018            47                                     2
2018            47                                     Total_T
2018            48                                     1
2018            48                                     2
2018            48                                     Total_T
2018            Total_W                                Total_T
Total_A         Total_W                                Total_T
```

Remarque 5 : La fonction DECODE a opéré sur les résultats de la fonction GROUPING. Dans cet exemple, DECODE retourne le texte Total_A, Total_W et Total_T si elle reçoit un 1 et que la valeur de Année, CodeWilaya et CodeType est à 0 respectivement.

- Temps d'exécution :

```
581 ligne(s) sélectionnée(s)

Ecoulé : 00 :00 :05.90
SQL>
```

Réponse 7

- Classement des spécialités dans chaque ville selon leurs montants versés :

- **Classement non dense :**

```
SQL> Select a.codeWilaya, a.codeBanque, sum(f.montantV) as MontantV,
  2         rank() over (order by sum(MontantV) Desc) as Classement
  3  from DAgence a, FOperation f
  4  where a.NumAgence = f.NumAgence
  5  group by (a.codeWilaya,a.codeBanque);

CODEWILAYA CODEBANQUE   MONTANTV CLASSEMENT
---------- ---------- ---------- ----------
        48          5 76734593,5          1
        30          7 74619637,9          2
        48          8 73428321,6          3
        30          2 68267672,6          4
        42          5 67717906,5          5
        30          3 67244632,6          6
        33          8 67203659,7          7
        30          5   66381123          8
        11          1 65455376,8          9
        33          1 65197027,1         10
```

- **Temps d'exécution :**

```
480 ligne(s) sélectionnée(s).

Ecoulé : 00 :00 :05.02
SQL>
```

Remarque 6.1 : La fonction RANK est une fonction de classement qui a permis d'attribuer pour chaque enregistrement une position par rapport aux autres enregistrements. mais cette fonction risque de renvoyer des trous dans un tel classement.

- **Classement dense :**

```
SQL> Select a.codeWilaya, a.codeBanque, sum(f.montantV) as MontantV,
  2        dense_rank() over (order by sum(MontantV) Desc) as Classement
  3  from DAgence a, FOperation f
  4  where a.NumAgence = f.NumAgence
  5  group by (a.codeWilaya,a.codeBanque);

CODEWILAYA CODEBANQUE   MONTANTV CLASSEMENT
---------- ---------- ---------- ----------
        48          5 76734593,5          1
        30          7 74619637,9          2
        48          8 73428321,6          3
        30          2 68267672,6          4
        42          5 67717906,5          5
        30          3 67244632,6          6
        33          8 67203659,7          7
        30          5   66381123          8
        11          1 65455376,8          9
        33          1 65197027,1         10
        30          4 63457582,1         11
```

Remarque 6.2 : Cette variante (DENSE_RANK) affecte à chaque enregistrement un indice en fonction de sa position en laissant aucun trou dans la séquence de classement.

- **Temps d'exécution :**

```
480 ligne(s) sélectionnée(s).

Ecoulé : 00 :00 :04.16
SQL>
```

Réponse 8

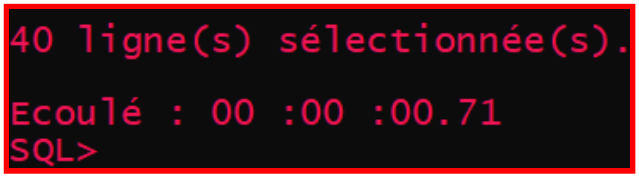
- Répartition cumulative du nombre d'opérations, par banque dans chaque année :

```
SQL> Select t.Année , d.NomBanque, sum(f.NbOperationR+f.NbOperationR) as SommeNb,
  2    cume_dist() over (partition by t.Année order by sum(f.NbOperationR+f.NbOperationR))
  3      as cum_dist_nb
  4  from FOperation f, DAgence d, DTemps t
  5  where f.NumAgence = d.NumAgence
  6  and f.CodeTemps = t.CodeTemps
  7  group by t.Année , d.NomBanque
  8  order by t.Année , d.NomBanque;

ANNÉE           NOMBANQUE                                          SOMMENB CUM_DIST_NB
--------------- -------------------------------------------------- ---------- -----------
2015            BNA 1                                               15058          ,9
2015            BNA 10                                              12764          ,1
2015            BNA 2                                               14146          ,3
2015            BNA 3                                               14566          ,4
2015            BNA 4                                               15326           1
2015            BNA 5                                               14844          ,8
2015            BNA 6                                               13986          ,2
2015            BNA 7                                               14688          ,6
2015            BNA 8                                               14616          ,5
2015            BNA 9                                               14762          ,7
```

Remarque 7 : Cette requête montre la répartition cumulative du nombre d'opérations par banque dans chaque année et ceci via la fonction CUME_DIST qui calcule la position du nombre d'opération par rapport aux banques dans chaque année dans l'ordre ascendant (ce rapport est défini par la clause PARTITION BY).

- **Temps d'exécution :**



Réponse 9

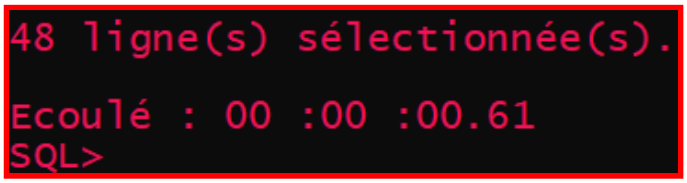
- Le nombre d'opérations global pour chaque mois, et segmentation des mois en 4 segments à l'aide de la fonction ntile :

```
SQL> select t.Mois, sum(f.NbOperationR+f.NbOperationV) as SommeNb,
  2    ntile(4) over(order by sum(f.NbOperationR+f.NbOperationV))
  3      as ntile_4
  4  from FOperation f, DTemps t
  5  where f.CodeTemps = t.CodeTemps
  6  group by t.Mois
  7  order by t.Mois;

MOIS               SOMMENB    NTILE_4
--------------- ---------- ----------
01/2015              13119          4
01/2016              12880          3
01/2017              12847          2
01/2018              13112          4
02/2015              11540          1
02/2016              12126          1
02/2017              11859          1
02/2018              11870          1
03/2015              12940          3
```

Remarque 8 : Cette requête attribue pour chaque nombre d'opération global par mois dans 4 buckets qui représentent les 4 segments des mois. La fonction NTILE effectue cette segmentation en affectant un numéro à chaque ligne de la partition.

- **Temps d'exécution :**



Réponse 10

- Ratio du montant versé pour chaque banque, dans chaque année :

```
SQL> Select t.Année, d.NomBanque, sum(f.MontantV) as MontV,
  2          sum(sum(f.MontantV)) over() as Total,
  3          ratio_to_report (sum(f.MontantV))
  4            over(partition by t.Année) as Ratio
  5  from FOperation f, DAgence d, Dtemps t
  6  where f.NumAgence = d.NumAgence and f.CodeTemps = t.CodeTemps
  7  group by t.Année, d.NomBanque;

ANNÉE           NOMBANQUE                                            MONTV      TOTAL      RATIO
--------------- -------------------------------------------------- ---------- ---------- ----------
2015            BNA 6                                              399611341 1,6887E+10 ,094549855
2015            BNA 9                                              434639137 1,6887E+10 ,102837591
2015            BNA 4                                              436084779 1,6887E+10 ,103179636
2015            BNA 8                                              430407567 1,6887E+10 ,101836382
2015            BNA 5                                              430033528 1,6887E+10 ,101747883
2015            BNA 10                                             384732575 1,6887E+10 ,091029472
2015            BNA 3                                              432210212 1,6887E+10 ,102262896
2015            BNA 2                                              426610351 1,6887E+10 ,100937944
2015            BNA 7                                              421472449 1,6887E+10 ,099722293
2015            BNA 1                                              430659745 1,6887E+10 ,101896048
```

Remarque 9 : Dans cette requête, la fonction RATIO_TO_REPORT calcule le ratio du MontantV par rapport à la somme des MontantV d'une année (cette information sur l'année est définie par la clause PARTITION BY).

- Temps d'exécution :

```
40 ligne(s) sélectionnée(s).

Ecoulé : 00 :00 :00.67
SQL>
```

Réponse 11

- L'agence qui réalise un nombre d'opérations maximal pour chaque banque :

```
SQL> Select codeBanque, NumAgence, nbOp
  2  from (select da.codeBanque, da.NumAgence,
  3          sum(fo.NbOperationR+fo.NbOperationV) as nbOp,
  4          max (sum(fo.NbOperationR+fo.NbOperationV))
  5            over (partition by da.codeBanque) as Max_nb_op
  6        from DAgence da, FOperation fo
  7  where da.NumAgence = fo.NumAgence
  8  group by da.codeBanque, da.NumAgence)
  9  where nbOp = Max_nb_op;

CODEBANQUE  NUMAGENCE       NBOP
---------- ---------- ----------
         1      10834         93
         2       5139         96
         2      11597         96
         2      12211         96
         3         92         91
         3       2174         91
         4      12224        111
         5      11109         92
         6       1553        110
         7       9518        100
         8       6196        104

CODEBANQUE  NUMAGENCE       NBOP
---------- ---------- ----------
         9      11394        103
        10      12284         99

13 ligne(s) sélectionnée(s).
```

Remarque 10 : Dans cette requête, la valeur agrégée MAX pour les deux colonnes (NbOperationR et NbOperationV) est calculée dans une partition (Banque) et utilisée pour comparer le nombre d'opération de chaque agence par rapport à la valeur max de sa propre partition (la banque à laquelle elle appartient).

- Temps d'exécution :

```
13 ligne(s) sélectionnée(s).

Ecoulé : 00 :00 :01.30
SQL>
```

Conclusion

Après la réalisation de ce TP, nous avons mieux compris les fonctions analytiques qui constituent un mécanisme puissant permettant de représenter de manière très simple des opérations analytiques complexes.