



SAE 2.04

EXPLOITATION D’UNE BDD

DECLERCQ Tifaine

GOLLIOT Camille

BUT 1 - GROUPE D

ANNÉE 2023-2024

SOMMAIRE

[**1 MCD**](#_ilqa333gxa44) **2**

[**2 EXERCICES**](#_q2cqsp6tbu4) **3**

[2.1 EXERCICE 1 - comprendre les données](#_fyfrvajut1vy) 3

[Q1 ANALYSE DU FICHIER RÉCUPÉRÉ](#_nvugxhii3w82) 3

[1.](#_o0jszl2596w3) 3

[2.](#_cr0jlsfcuj8d) 3

[3.](#_mtl4rmudtfnw) 3

[4.](#_ui8hnbd4h8q7) 4

[5.](#_k3w791st5t0) 4

[6.](#_xv6nm7f9xv2s) 4

[7.](#_yo8fbku2sbq4) 5

[8.](#_1kvf89qx1zxp) 5

[9.](#_vmdazd1u3mmm) 5

[Q2 IMPORTER LES DONNÉES](#_54sith2apvlw) 6

[1.](#_bfbh0ofgbnch) 6

[2.](#_vunq5nqmh6xb) 6

[3.](#_fklx3tpjbg0) 7

[4.](#_5wjt0su5qwlp) 7

[5.](#_w67enbmn5gbq) 7

[2.2 EXERCICE 2](#_ng9nzay6pfon) 8

[Q1 NORMALISATION DES DONNÉES](#_mmsosn5r296q) 8

[1.](#_x40r9n6ot4om) 8

[2.](#_fbq3wilpy5sn) 8

[Q2 UNE QUESTION DE TAILLE](#_2y6kgui4c7mu) 8

[1.](#_69iiabgb7279) 8

[2.](#_803r7rbg5pto) 8

[3.](#_z4l44m1o95tn) 8

[4.](#_9htyyu6w66nx) 9

[2.3 EXERCICE 3](#_4d9iurncosah) 10

[Q1.](#_43m645ui5dr2) 10

[Q2.](#_9m4hv4ons081) 10

[Q3.Q4.](#_i3tif9gswx1h) 11

[Q5.](#_52lbt5t0x9ju) 14

[Q6.](#_6i80xff6eklm) 15

[Q7.](#_2h517vj5rmsh) 15

[Q8.](#_tjqbp8b4rk0u) 16

# 1 MCD

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 2 EXERCICES

## 2.1 EXERCICE 1 - comprendre les données

### **Q1 ANALYSE DU FICHIER RÉCUPÉRÉ**

#### **1.**

Nous devons déterminer le nombre de lignes que contient le fichier, pour cela sous linux nous allons faire :

| wc -l <nom du fichier> |
| --- |

Dans notre cas le fichier s' appelle : “fr-esr-parcoursup.csv” on aura donc comme commande :

| wc -l fr-esr-parcoursup.csv |
| --- |

on obtient ce résultat : 

ce qui nous donne donc :

| **Nombre de lignes ( avec entête)** | **Nombre de lignes (sans entête)** |
| --- | --- |
| 13870 | 3869 |

#### **2.**

Une ligne représente une formation dans un établissement pour une année avec ses différentes informations (nombre de places disponibles, localisation …)

#### **3.**

Nous devons trouver combien de colonnes contient notre fichier, pour cela nous devons utiliser plusieurs commandes:

| head -n <nom du fichier> | sed ‘s/[^;]//g |wc -c |
| --- |

Cette commande nous permet de ne garder que la première ligne du fichier, pour en remplacer les ‘;’ par une chaîne vide pour ne garder que les délimiteur (‘;’) pour en compter le nombre, nous allons donc avoir une commande qui ressemble à cela :

| head -1 fr-esr-parcoursup.csv | sed 's/[^;]//g' | wc -c |
| --- |

nous obtenons ce résultat : 

Il y a donc 118 colonnes dans le fichier CSV.

#### **4.**

Il y a deux colonnes qui identifient les établissements, une est plus précise que l' autre car elle contient le code UAI de l'établissement qui est un code unique.

| **Entête de la colonne** | **Numéro de la colonne** |
| --- | --- |
| Code UAI de l'établissement | 3 (colonne n3) |
| Établissement | 4 (colonne n4) |

#### **5.**

Il y a plusieurs colonnes qui servent à identifier une formation mais après plusieurs recherches la plus précise est la colonne 110, colonne qui a pour entête cod\_aff\_form qui correspond au code de la formation affiliée à l'établissement.

| **Entête de la colonne** | **Numéro de la colonne** |
| --- | --- |
| cod\_aff\_form | 110 (colonne n110) |

#### **6.**

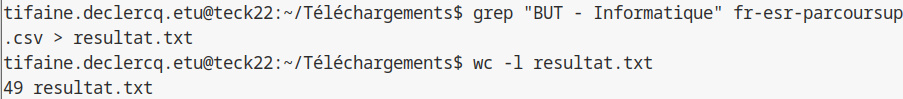
Pour trouver le nombre de ligne correspondant à notre BUT, nous allons d'abord isoler les lignes du fichier qui contiennent “BUT - Informatique” pour les mettre dans un fichier :

| grep "BUT - Informatique" fr-esr-parcoursup.csv > resultat.txt |
| --- |

nous avons compté le nombre de lignes que contient ce nouveau fichier avec :

| wc -l resultat.txt |
| --- |

et nous avons obtenu ce résultat :



il y a donc 49 BUT Informatique dans toute la france, pour aller plus loins nous avons regardé combien de BUT informatique se trouvent à Villeneuve d'Ascq avec :

| grep "Villeneuve" resultat.txt | wc -l |
| --- |

nous obtenons ce résultat :



Il y a donc une seule formation BUT Informatique à Villeneuve d'Ascq.

#### **7.**

Il y a deux colonnes qui identifient les départements :

| **Entête de la colonne** | **Numéro de la colonne** |
| --- | --- |
| Code départemental de l’établissement | 5 (colonne n 5) |
| Département de l’établissement | 6 (colonne n 6) |

#### **8.**

Nous allons créer une table import dans postgreSQL avec la commande :

| \copy import from parcoursup.csv with (format CSV, delimiter ';', HEADER); |
| --- |

#### 

#### **9.**

Nous avons rencontré une difficulté au niveau des valeurs null car au début nous cherchions à tout prix à les remplacer à la main avec :

| NULL AS 'NA' |
| --- |

mais au final le logiciel le fait de lui-même.

### **Q2 IMPORTER LES DONNÉES**

#### **1.**

cf dico.xls la première colonne correspond aux noms des colonnes dans le fichier csv, la deuxième colonne correspond aux noms des colonnes dans la table import et la troisième colonne correspond aux noms des colonnes dans les tables ventilées.

#### **2.**

Pour créer notre table nous avons utilisé cette commande dans postgreSQL:

| create table if not exists import(n1 integer default null, n2 Char(33) default null, n3 Char(8) default null, n4 text default null, n5 Char(3) default null, n6 Char(24) default null, n7 text default null, n8 text default null, n9 text default null, n10 text default null, n11 text default null, n12 text default null, n13 text default null, n14 text default null, n15 text default null, n16 text default null, n17 text default null, n18 text default null, n19 text default null, n20 text default null, n21 text default null, n22 text default null, n23 text default null, n24 text default null, n25 text default null, n26 text default null, n27 text default null, n28 integer default null, n29 integer default null, n30 integer default null, n31 integer default null, n32 integer default null, n33 integer default null, n34 integer default null, n35 integer default null, n36 integer default null, n37 integer default null, n38 integer default null, n39 integer default null, n40 integer default null, n41 integer default null, n42 integer default null, n43 integer default null, n44 integer default null, n45 integer default null, n46 integer default null, n47 float default null, n48 integer default null, n49 integer default null, n50 integer default null, n51 float default null, n52 float default null, n53 float default null, n54 float default null, n55 float default null, n56 float default null, n57 float default null, n58 float default null, n59 float default null, n60 integer default null, n61 integer default null, n62 integer default null, n63 integer default null, n64 integer default null, n65 integer default null, n66 float default null, n67 integer default null, n68 float default null, n69 integer default null, n70 integer default null, n71 integer default null, n72 integer default null, n73 integer default null, n74 float default null, n75 float default null, n76 float default null, n77 float default null, n78 float default null, n79 float default null, n80 float default null, n81 float default null, n82 float default null, n83 float default null, n84 float default null, n85 float default null, n86 float default null, n87 float default null, n88 float default null, n89 float default null, n90 float default null, n91 float default null, n92 float default null, n93 float default null, n94 float default null, n95 float default null, n96 float default null, n97 float default null, n98 float default null, n99 float default null, n100 float default null, n101 float default null, n102 Char(39) default null, n103 float default null, n104 Char(39) default null, n105 integer default null, n106 Char(39) default null, n107 integer default null, n108 Char(45) default null, n109 Char(19) default null, n110 integer default null, n111 text default null, n112 text default null, n113 float default null, n114 float default null, n115 float default null, n116 float default null, n117 Char(5) default null, n118 Char(5) default null); |
| --- |

#### **3.**

Cf Q2.2

#### **4.**

Une fois la table créée nous avons dû la remplir avec les données du fichier csv de Parcoursup:

| \copy import from <chemin/du/fichier.csv> with (format CSV, delimiter ';', HEADER); |
| --- |

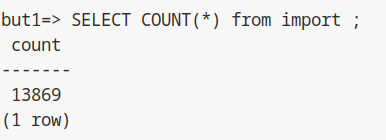
nous avons, changé <chemin/du/fichier.csv> par le chemin réel et le nom de notre fichier, étant donné que notre fichier se trouve là où nous avons lancé le serveur postgreSQL, notre commande est:

| \copy import from fr-esr-parcoursup.csv with (format CSV, delimiter ';', HEADER); |
| --- |

#### **5.**

Nous devons compter le nombre de formations qu’il y a dans le fichier, étant donné qu'une ligne dans le fichier correspond à une formation nous avons fait :

| SELECT COUNT(\*) from import ; |
| --- |

nous avons obtenu comme résultat : 

Il y a donc 13 869 formations dans Parcoursup pour l'année 2023.

## **2.2 EXERCICE 2**

### **Q1 NORMALISATION DES DONNÉES**

nous avons décidé de décomposer la table import en x différentes tables : la table

#### **1.**

cf page 2

#### **2.**

cf parcoursup.sql

### **Q2 UNE QUESTION DE TAILLE**

### **1.**

| wc -c fr-ser-parcoursup.csv |
| --- |

#### 

### **2.**

| SELECT pg\_total\_relation\_size('import') AS "Taille de la table"; |
| --- |

### 

#### **3.**

| SELECT pg\_total\_relation\_size('regions') + pg\_total\_relation\_size('departements') + pg\_total\_relation\_size('academies') + pg\_total\_relation\_size('etablissements') + pg\_total\_relation\_size('formations') + pg\_total\_relation\_size('effectifs') AS "Taille des tables de la ventilation"; |
| --- |

#### 

#### 

#### **4.**

d’abord on export les tables avec :

| pg\_dump -U postgres -t <nom de la table> -d <nom de la base> > fichierCible.sql  exemple pour la table departements :  pg\_dump -U postgres -t departements -d postgres > departements.sql |
| --- |

et ensuite on affiche leur taille avec :

| wc -c departements.sql |
| --- |

ce qui nous donne :





#### 

## 

## **2.3 EXERCICE 3**

### **Q1.**

la colonne n56 correspond à l'effectif total des néo bacheliers admis, pour la calculer nous devons donc additionner les colonnes 57, 58 et 59 pour cela nous avons fait ( nous avons limité le résultat de la requête à 15 lignes) :

| SELECT n56, n57 + n58 + n59 AS result FROM import LIMIT 15; |
| --- |

#### 

### **Q2.**

Pour s'assurer que le résultat est parfaitement exacte, nous pourrions vérifier les deux colonnes, ligne par ligne, mais ce serait trop long. Nous avons alors écrit une requête qui sélectionne la ligne 56, qui additionne les colonnes 57, 58 et 59 et qui stocke ce résultat dans une colonne result. Ensuite la requête compare la colonne 56 avec la colonne result et nous sort toutes les lignes qui ont des résultats différents dans les deux colonnes :

| SELECT n57 + n58 + n59 AS result, n56 FROM import WHERE n57 + n58 + n59 <> n56 LIMIT 15; |
| --- |

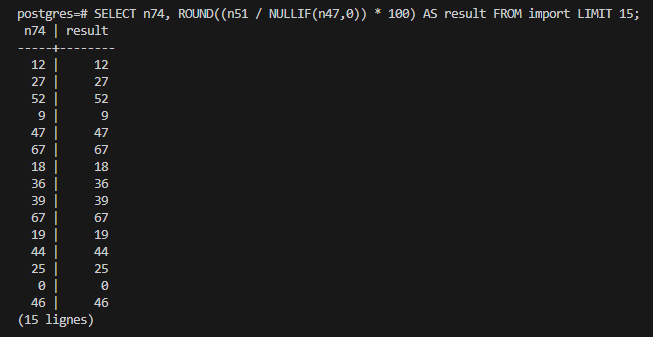
### 

### 

### **Q3.Q4.**

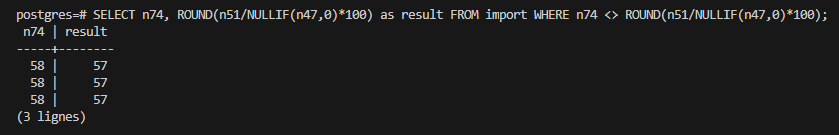
La colonne n 74 correspond à l'effectif des candidats en terminale générale ayant reçu une proposition d’admission de la part de l’établissement. Pour la calculer de nouveau, nous devons donc prendre la colonne n 51 qui correspond à la part effectif des admis ayant reçu leur proposition d’admission à l'ouverture de la procédure principale que l'on va diviser par la colonne n 47 qui correspond à l’effectif total des candidats ayant accepté la proposition de l’établissement (admis) ce qui nous donne donc :

| SELECT n74, ROUND((n51 / NULLIF(n47,0)) \* 100) AS result FROM import LIMIT 15; |
| --- |

****

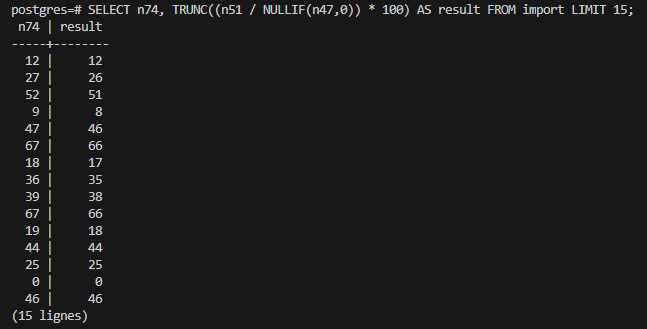
Pour vérifier, nous cherchons les valeurs de la colonne result qui sont différentes de celles dans la colonne 74 ce qui nous donne :

| SELECT n74, ROUND(n51/NULLIF(n47,0)\*100) as result FROM import WHERE n74 <> ROUND(n51/NULLIF(n47,0)\*100); |
| --- |



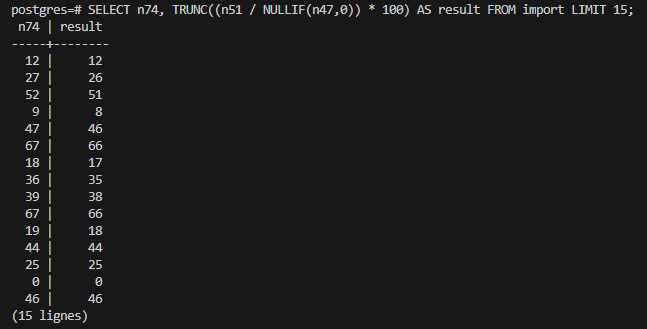
Nous constatons que 3 lignes ne donnent pas le résultat attendu, nous avons donc essayé une autre méthode : TRUNC

| SELECT n74, TRUNC((n51 / NULLIF(n47,0)) \* 100) AS result FROM import LIMIT 15; |
| --- |



Pour vérifier, nous exécutons la même requête qu'avant avec les modifications nécessaires :

| SELECT n74, TRUNC(n51/NULLIF(n47,0)\*100) as result FROM import WHERE n74 <> TRUNC(n51/NULLIF(n47,0)\*100); |
| --- |

ce qui nous donne : 

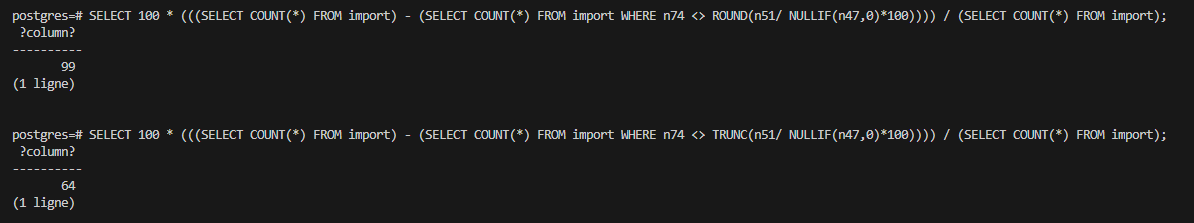
Nous constatons qu'une méthode est plus fiable qu'une autre. Pour en être sûr, nous avons calculé le pourcentage de réussite du calcul de chacune des méthodes.

ROUND:

| SELECT 100 \* (((SELECT COUNT(\*) FROM import) - (SELECT COUNT(\*) FROM import WHERE n74 <> ROUND(n51/ NULLIF(n47,0)\*100)))) / (SELECT COUNT(\*) FROM import); |
| --- |

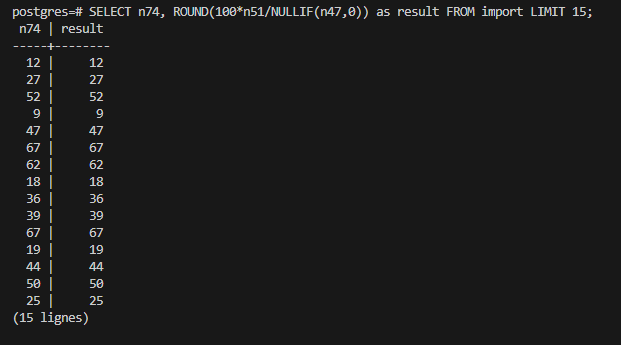
TRUNC:

| SELECT 100 \* (((SELECT COUNT(\*) FROM import) - (SELECT COUNT(\*) FROM import WHERE n74 <> TRUNC(n51/ NULLIF(n47,0)\*100)))) / (SELECT COUNT(\*) FROM import); |
| --- |



Nous pouvons donc en conclure que la méthode la plus fiable est ROUND mais en continuant de chercher nous nous sommes aperçues que le problème résidait dans la précision de postgresql quant aux calculs de petits résultats. Pour résoudre ce problème, nous avons décidé de diviser par un plus grand nombre grâce à cette relation mathématique : en pratique, au lieu de diviser le résultat par 100 pour avoir le pourcentage nous commençons par multiplier l'effectif par 100 avant de diviser par le total ainsi on manipule de grands nombres ce qui nous donne :

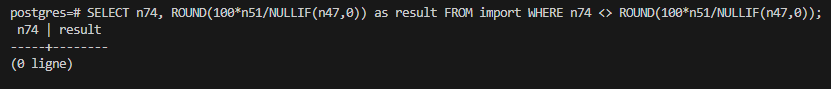
| SELECT n74, ROUND(100\*n51/NULLIF(n47,0)) as result FROM import LIMIT 15; |
| --- |

****

Nous obtenons en apparence un meilleur résultat pour en être sûr, nous exécutons la même requête de vérification avec les modifications adéquates :

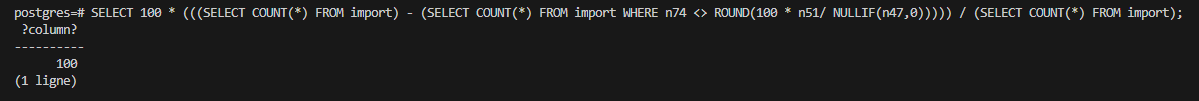
| SELECT n74, ROUND(100\*n51/NULLIF(n47,0)) as result FROM import LIMIT 15; |
| --- |

ce qui nous donne :



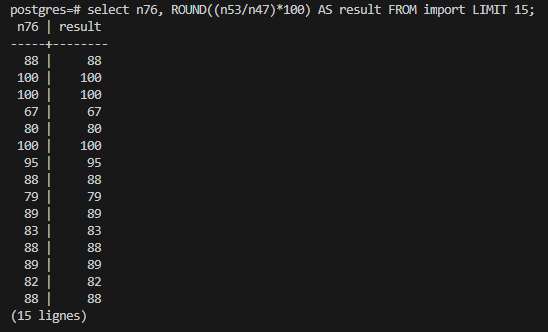
C’est donc la solution la plus adéquate avec un pourcentage de réussite de 100% :

| SELECT 100 \* (((SELECT COUNT(\*) FROM import) - (SELECT COUNT(\*) FROM import WHERE n74 <> ROUND(100 \* n51/ NULLIF(n47,0))))) / (SELECT COUNT(\*) FROM import); |
| --- |

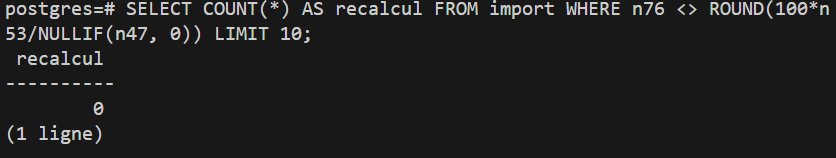


### **Q5.**

| SELECT n76, ROUND((n53/n47)\*100) AS result FROM import LIMIT 15. |
| --- |

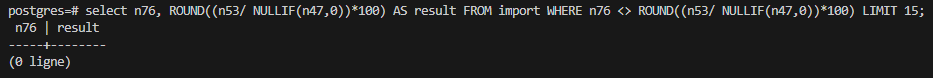


| SELECT COUNT(\*) AS recalcul FROM import WHERE n76 <> ROUND(100\*n53/NULLIF(n47, 0)) LIMIT 10; |
| --- |



### **Q6.**

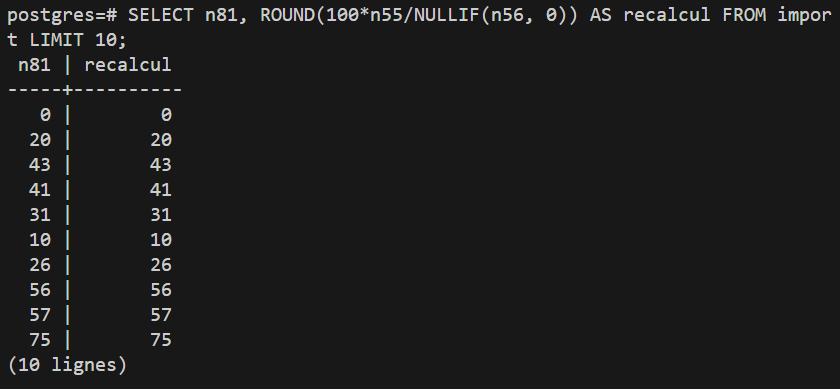
| SELECT n76, ROUND((n53/ NULLIF(n47,0))\*100) AS result FROM import WHERE n76 <> ROUND((n53/ NULLIF(n47,0))\*100) LIMIT 15; |
| --- |



### **Q7.**

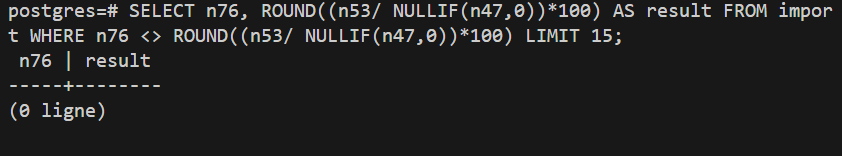
Pour recalculer n81, nous devons diviser n55 par n56 sans oublier de multiplier n55 par 100 pour les raisons vu précédemment ce qui nous donne la requête suivante :

| SELECT n81, ROUND(100\*n55/NULLIF(n56, 0)) AS recalcul FROM import LIMIT 10; |
| --- |



On vérifie qu'il n’y ai pas de résultats différents entres les deux colonnes :

| SELECT n76, ROUND((n53/ NULLIF(n47,0))\*100) AS result FROM import WHERE n76 <> ROUND((n53/ NULLIF(n47,0))\*100) LIMIT 15; |
| --- |



### 

### **Q8.**

On obtient le même résultat dans les tables ventilées.

| SELECT ROUND(100\*nbNeoBacBoursierAdmis/NULLIF((nbNeoBacGeneral+nbNeoBacTechno+nbNeoBacPro), 0)) AS n81 FROM effectifs LIMIT 10; |
| --- |

# 