

# TD 4 : SQL

## Aggrégations, Partitions, Fenêtres

2024-10-11

 Avec solutions

---

L3 MIASHS/Ingémath  
Université Paris Cité  
Année 2024  
[Course Homepage](#)  
[Moodle](#)



[Documentation Postgres](#)

[Documentation Postgres en Français](#)

## Fonctions d'agrégation

Les fonctions d'*agrégation* permettent d'effectuer des opérations avancées sur les solutions d'une requête (sur une table) comme : compter les lignes, sélectionner le maximum dans une colonne, etc.

Une des opérations les plus courantes est de compter. `COUNT(col)` permet de compter les résultats d'une requête.

### Count

Pour compter les pays en Europe, on écrira :

```
SELECT COUNT(countrycode)
FROM world.country
WHERE continent='Europe';
```

Cette requête renvoie une table ayant *une* ligne et *une* colonne contenant le nombre de lignes dans le résultat de la requête.

## GROUP BY

Admettons qu'on veuille compter les pays par continent. On serait tenté d'écrire :

```
SELECT
  continent, COUNT(countrycode)
FROM
  world.country;
```

Cependant, cette requête ne fonctionnera pas en SQL. SQL est incapable de deviner comment regrouper les différentes lignes pour compter. On doit lui spécifier clairement cela avec la clause `GROUP BY` :

```
SELECT
  continent, COUNT(countrycode)
FROM
  world.country
GROUP BY
  continent;
```

Cette requête regroupe les lignes de la table `country` par modalité de la colonne `continent` et pour chaque groupe compte le nombre de `countrycode` y apparaissant. Lorsque plusieurs lignes sont susceptibles d'avoir la même valeur, on peut compter seulement le nombre d'occurrences distinctes avec `COUNT(DISTINCT col)`.

**i** Question

Écrire une requête qui compte le nombre de langues parlées dans chaque pays.

**💡** Solution

```
SELECT
  countrycode, COUNT(language)
FROM
  world.countrylanguage
GROUP BY
  countrycode;
```

**i** Question

Écrire une requête qui compte le nombre de langues parlées dans le monde.

**💡** Solution

```
SELECT
  COUNT(DISTINCT language)
FROM
  world.countrylanguage;
```

**i** Question

Écrire une requête qui compte le nombre de langues officielles par pays.

Une solution presque bonne :

**💡** Solution

```
SELECT countrycode, COUNT(language)
FROM world.countrylanguage
WHERE isofficial
GROUP BY countrycode;
```

Cependant, on rate les pays qui ne possèdent pas de langue officielle. On va utiliser une superbe jointure extérieure :

### Solution

```
SELECT P.countrycode, COUNT(language)
FROM
  world.country as P
LEFT JOIN
  world.countrylanguage as L
  ON (P.countrycode = L.countrycode and L.isofficial)
GROUP BY P.countrycode;
```

## Sum, Max, Min, Avg

Une autre fonction importante est la fonction `SUM(col)` qui effectue la somme des valeurs (numériques) d'une colonne :

```
SELECT SUM(population_country)
FROM world.country;
```

renvoie la population mondiale.

On peut de même utiliser `GROUP BY` pour faire des paquets :

```
SELECT
  continent, SUM(population_country)
FROM
  world.country
GROUP BY continent;
```

renvoie la population de chaque continent.

On peut même faire des opérations sur la colonne à l'intérieur de `SUM`. Par exemple : `SUM(percentage/100)`.

### Question

Écrire une requête qui renvoie le nombre de langues officielles par pays

### Solution

```
WITH s AS(
  SELECT L.countrycode, sum(CAST (isofficial AS INTEGER)) AS n_official
  FROM world.countrylanguage as L
  GROUP BY L.countrycode)
SELECT s.*, c.name_country
FROM world.country c NATURAL JOIN s
ORDER BY s.n_official DESC, s.countrycode;
```

## Requêtes (I)

### Question

Écrire une requête qui renvoie la surface de chaque région.

 **Solution**

```
SELECT region, SUM(surfacearea)
FROM world.country
GROUP BY region;
```

 **Question**

Écrire une requête qui compte le nombre de francophones dans le monde.

 **Solution**

```
SELECT
  SUM((percentage/100)*population_country)
FROM
  world.country
NATURAL JOIN
  world.countrylanguage
WHERE
  language = 'French';
```

On peut utiliser de la même façon la fonction MIN (resp. MAX) qui renvoie la plus petite (resp. grande) valeur ou AVG qui renvoie la moyenne.

 **Question**

Combien de personnes vivent dans une capitale européenne ?

 **Solution**

```
SELECT
  SUM(ci.population)
FROM
  world.country co
JOIN
  world.city ci ON co.capital=ci.id
WHERE
  co.continent='Europe';
```

 **Question**

Quelle est la capitale européenne la moins peuplée ?

### Solution

```
WITH r as (  
  select ci.name, ci.population as s  
  FROM  
  world.city ci  
  JOIN  
  world.country co ON co.capital = ci.id  
  WHERE  
  co.continent='Europe'  
)  
SELECT  
  r.name  
FROM  
  r  
WHERE r.s = (  
  SELECT min(r.s)  
  FROM r)  
;
```

### Question

Quelle est la langue la plus parlée dans le monde ?

### Solution

```
WITH R AS (  
  SELECT language, SUM((percentage/100)*population_country) as s  
  FROM  
  world.country  
  NATURAL JOIN  
  world.countrylanguage  
  GROUP BY language  
)  
  
SELECT  
  language  
FROM  
  R  
WHERE s = (  
  SELECT MAX(s)  
  FROM R  
)  
);
```

ou plus simplement

```
SELECT  
  language, SUM((percentage/100)*population_country) as s  
FROM  
  world.country  
NATURAL JOIN  
  world.countrylanguage  
GROUP BY language  
ORDER BY s DESC  
LIMIT 1 ;
```

## Having

Parfois, on veut filtrer les requêtes en fonction du résultat d'une fonction d'agrégation.

Par exemple, pour connaître les langues officielles dans plus de 10 pays, on serait tenté d'écrire :

```
SELECT
  language
FROM
  world.countrylanguage
WHERE
  COUNT(countrycode) > 10 AND isofficial
GROUP BY language;
```

🔥 Cela ne fonctionne pas. **WHERE** applique une condition sur chaque ligne de la table pour les filtrer, par exemple, garder seulement les langues officielles. Ici, on veut *ensuite* sélectionner les lignes après avoir regroupé par langue et compté.

On utilisera alors **HAVING**, après la clause **GROUP BY** :

```
SELECT
  language
FROM
  world.countrylanguage
WHERE
  isofficial
GROUP BY language
HAVING
  COUNT(countrycode) > 10;
```

- ① La requête concerne la table `world.countrylanguage`
- ② On filtre les lignes qui correspondent à des langues officielles
- ③ On groupe/partitionne la table filtrée selon la langue
- ④ On ne garde que les groupes comportant au moins 10 tuples
- ⑤ On projette le résultat sur la colonne `language`

## Requêtes (II)

- Écrire une requête qui renvoie le nombre de pays par régime.

### 💡 Solution

```
SELECT
  governmentform AS regime, COUNT(countrycode) AS nombre
FROM
  world.country
GROUP BY governmentform
ORDER BY governmentform ;
```

### i Question

Écrire une requête calculant le nombre de personnes vivant dans des villes de plus d'un million d'habitants.

 **Solution**

```
SELECT
  SUM(population) AS pop
FROM
  world.city
WHERE
  population >= 1000000 ;
```

 **Question**

Écrire une requête qui calcule le nombre total de personnes vivants dans des villes qui ne sont pas listées dans la table `city`. (Indice : comparer la population du pays avec la somme sur les villes).

 **Solution**

```
WITH
  pop_villes AS (
    SELECT
      SUM(population) AS pop
    FROM
      world.city),
  pop_totale AS (
    SELECT
      SUM(population_country) AS pop
    FROM
      world.country)

SELECT
  pop_totale.pop - pop_villes.pop
FROM
  pop_totale, pop_villes ;
```

Réponse : (4,649,189,566)

 **Question**

Écrire une requête qui compte le nombre moyen de langues parlées par pays dans chaque région.

### Solution

```
WITH nb_langues_parlees AS (  
  SELECT  
    countrycode, COUNT(language) nb  
  FROM  
    world.countrylanguage  
  GROUP BY countrycode  
)  
  
SELECT  
  region, SUM(nb)/COUNT(DISTINCT countrycode)  
FROM  
  world.country co  
NATURAL JOIN  
  nb_langues_parlees  
GROUP BY region ;
```

Avec la fonction AVG()

```
WITH nb_langues_parlees AS (  
  SELECT  
    countrycode, COUNT(language) nb  
  FROM  
    world.countrylanguage  
  GROUP BY countrycode  
)  
  
SELECT  
  region, AVG(nb)  
FROM  
  world.country co  
NATURAL JOIN  
  nb_langues_parlees  
GROUP BY region ;
```

### Question

Écrire une requête qui donne la liste des pays ayant deux langues officielles parlées par plus de la moitié de la population.

👉 Pas besoin d'agrégation à cet endroit là.

**i** Solution

```
WITH R AS (  
  SELECT *  
  FROM  
    countrylanguage  
  WHERE  
    isofficial AND percentage >= 25  
)  
  
SELECT  
  R1.countrycode, R1.language, R2.language  
FROM  
  R AS R1  
JOIN  
  R AS R2 ON (  
    R1.countrycode=R2.countrycode AND  
    R1.language < R2.language  
  ) ;
```

**i** Question

Écrire une fonction `plus_peuplee(p_name_country text)` qui, étant donné le nom d'un pays, renvoie le nom de la ville la plus peuplée de ce pays. (schéma : `world`)

### Solution

```
CREATE FUNCTION public.plus_peuplee(p_name_country text)
RETURNS text
LANGUAGE sql AS $$
WITH
  R AS (
    SELECT
      co.countrycode
    FROM
      world.country co
    WHERE
      co.name_country=p_name_country
  ),
  M AS (
    SELECT
      MAX(population) AS m_pop
    FROM
      world.city ci
    WHERE
      ci.countrycode IN (
        SELECT
          countrycode
        FROM
          R
      )
  )
SELECT
  ci.name
FROM
  world.city ci
WHERE
  ci.countrycode IN (
    SELECT
      countrycode
    FROM
      R
  ) AND
  ci.population >= (
    SELECT
      m_pop
    FROM
      M
  ) ;
$$ ;
```

### Question

Écrire une fonction `langues_region(p_continent text)` qui étant donné le nom d'un continent, renvoie le nombre moyen de langues parlées par pays dans chaque région (schéma : `world`). L'entête de cette fonction doit être :

```
FUNCTION langues_region(p_continent TEXT)
RETURNS TABLE(region TEXT, nbmoy NUMERIC)
```

### Solution

```
CREATE FUNCTION public.languages_region(p_continent TEXT)
RETURNS TABLE(region TEXT, nbmoy NUMERIC)
LANGUAGE SQL AS $$
WITH S AS (
  SELECT
    R.region, cl.countrycode, COUNT(cl.language) AS n_l
  FROM
    (
      SELECT
        co.region, co.countrycode
      FROM
        world.country co
      WHERE
        co.continent = 'Asia'
    ) AS R
  JOIN
    world.countrylanguage cl USING (countrycode)
  GROUP BY
    R.region, cl.countrycode
)
SELECT
  region, AVG(n_l) AS nbmoy
FROM
  S
GROUP BY
  region ;
$$ ;
```

### Question

Ecrire une fonction `actor_category(p_nom text, p_prenom text)` qui prend en argument le nom et le prénom d'un acteur (d'une actrice) et renvoie la liste des noms des catégories de films dans lesquels il/elle a joué (schéma : `pagila`).

### Solution

```
CREATE FUNCTION public.actor_category(p_nom text, p_prenom text)
RETURNS text
LANGUAGE SQL AS
$$
SELECT
  string_agg(DISTINCT pc.name, ', ')
FROM
  (SELECT
    actor_id
  FROM
    pagila.actor
  WHERE
    last_name = p_nom AND
    first_name = p_prenom
  ) as pa
JOIN
  pagila.film_actor as pfa USING(actor_id)
JOIN
  pagila.film_category as pfc USING(film_id)
JOIN
  pagila.category as pc USING(category_id) ;
$$ ;
```

```
bd_2023-24=# SELECT public.actor_category('LOLLOBRIGIDA', 'JOHNNY') ;
               actor_category
```

-----  
Action, Animation, Children, Comedy, Documentary, Drama, Games, Horror, Music, New, Sci-Fi, S  
(1 row)

### Question

Ecrire une fonction `co_actors(p_nom text, p_prenom text)` qui renvoie les noms et prénoms des acteurs qui jouent dans un film où apparaît un acteur ou une actrice dont le nom et le prénom sont donnés en argument (schéma : `pagila`).

### Solution

```
CREATE OR REPLACE FUNCTION public.co_actors(p_nom text, p_prenom text)
RETURNS TABLE(nom text, prenom text)
LANGUAGE SQL AS
$$
SELECT
  DISTINCT pa2.last_name as nom, pa2.first_name as prenom
FROM
  (SELECT
    actor_id
  FROM
    pagila.actor
  WHERE
    last_name = p_nom AND
    first_name = p_prenom
  ) as pa
JOIN
  pagila.film_actor as pfa1 USING(actor_id)
JOIN
  pagila.film_actor as pfa2 ON (
    pfa1.film_id=pfa2.film_id AND
    pfa1.actor_id<>pfa2.actor_id
  )
JOIN
  pagila.actor as pa2 ON (pa2.actor_id=pfa2.actor_id)
ORDER BY nom, prenom;
$$ ;
```

### Question

Écrire une vue qui contient une ligne pour chaque pays où on parle français, présente les pays par population croissante, et contient trois colonnes :

- `name_country` (même type que dans `world.country`);
- `cumul_loc` (de type `float4`) qui donne le nombre cumulé de locuteurs du français dans les pays où on parle français, pas plus peuplés que le pays courant ;
- `cum_pop` (de type `float4`) qui donne la population cumulée des pays où on parle français, pas plus peuplés que le pays courant.

 Utilisez une fenêtre (`WINDOW`) avec une clause `RANGE` . . . .

 Pour trouver les pays où on parle français, utilisez l'expression `language like '%French%'`. Vous remarquerez que dans certains pays, il existe plusieurs variétés de 'French'. Veillez à compter tous les locuteurs, et à ne compter les habitants qu'une seule fois.

### Solution

```
WITH f AS (  
  SELECT  
    cl.countrycode, SUM(cl.percentage) AS percentage  
  FROM  
    world.countrylanguage cl  
  WHERE  
    cl.language LIKE '%French%'  
  GROUP BY  
    cl.countrycode  
)  
  
SELECT  
  co.name_country,  
  SUM(f.percentage * co.population_country::float4/100) OVER w AS cumul_loc,  
  SUM(co.population_country::float4) OVER w AS cum_pop  #<1>  
FROM  
  f  
NATURAL JOIN  
  world.country co  
WINDOW w AS (  
  ORDER BY co.population_country  
  RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW  
);
```

1. Pose problème!!! Si plusieurs formes de français sont utilisées dans un pays, la population de ce pays est comptabilisée plusieurs fois.