



# Bases de données : (encore) un peu plus loin dans SQL

Karèn Fort (repris d'Alice Millour)

[karen.fort@sorbonne-universite.fr](mailto:karen.fort@sorbonne-universite.fr)



## Sources d'inspiration

- ▶ il s'agit du cours d'Alice Millour (2020)
- ▶ <https://openclassrooms.com/fr/courses/1959476-administrez-vos-bases-de-donnees-avec-mysql/1966846-fonctions-dagregation>
- ▶ <https://openclassrooms.com/fr/courses/4449026-initiez-vous-a-lalgebre-relationnelle-avec-le-langage-sql/4558491-ameliorez-vos-agregations-grace-a-having>
- ▶ <https://sql.sh/cours/alias>

## Dans ce cours

- ▶ aller plus loin dans le `SELECT` : compter les éléments, trouver le minimum ou calculer la moyenne d'une colonne, etc.
- ▶ grouper les lignes qui partagent une même valeur pour faire des statistiques plus fines sur les données

## Agréger les résultats

Filtrer les résultats après une agrégation

Les alias

Pour finir

# L'agrégation

- ▶ Requête sans agrégation : renvoie une **liste** = liste des lignes sélectionnées par la requête
- ▶ Requête avec agrégation : renvoie une **valeur** = résultat d'une fonction d'agrégation appliquée à une colonne

## Fonction d'agrégation

permet (principalement) de calculer des statistiques sur les données :

- ▶ **COUNT** (colonne) = compter les éléments **not NULL**
- ▶ **MIN** (colonne), **MAX** (colonne) = trouver les éléments minimum et maximum d'une colonne
- ▶ **SUM** (colonne) = sommer les éléments d'une colonne
- ▶ **AVG** (colonne) = calculer la valeur moyenne des éléments d'une colonne

# (rappel) requête sans agrégation

Dans la BD Enquete

**SELECT** filiere **FROM** Diplomes

✓ Affichage des lignes 0 - 24 (total de 43, traitement en 0.0007 seconde(s).)

```
select filiere from Diplomes
```

Profilage [Éditer en ligne]

1 > >> |  Tout afficher | Nombre de lignes : 25 | Filtrer le:

+ Options

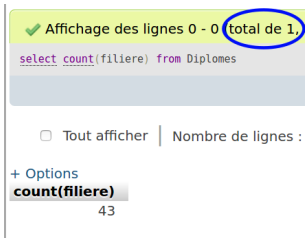
				filiere
<input type="checkbox"/>	Éditer	Copier	Supprimer	Informatique
<input type="checkbox"/>	Éditer	Copier	Supprimer	Informatique
<input type="checkbox"/>	Éditer	Copier	Supprimer	Informatique
<input type="checkbox"/>	Éditer	Copier	Supprimer	Informatique
<input type="checkbox"/>	Éditer	Copier	Supprimer	Sciences pour l'ingenieur
<input type="checkbox"/>	Éditer	Copier	Supprimer	Sciences pour l'ingenieur
<input type="checkbox"/>	Éditer	Copier	Supprimer	Sciences pour l'ingenieur
<input type="checkbox"/>	Éditer	Copier	Supprimer	Sciences pour l'ingenieur
<input type="checkbox"/>	Éditer	Copier	Supprimer	Sciences de l'education
<input type="checkbox"/>	Éditer	Copier	Supprimer	Sciences de l'education
<input type="checkbox"/>	Éditer	Copier	Supprimer	Sciences de l'education

# Requête avec agrégation : COUNT (colonne)

Dans la BD Enquete

SELECT COUNT (filiere) FROM Diplomes

43 filières



✓ Affichage des lignes 0 - 0 (total de 1)

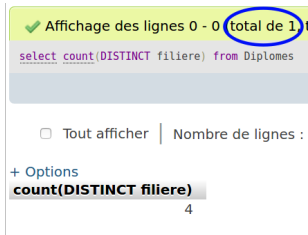
```
select count(filiere) from Diplomes
```

Tout afficher | Nombre de lignes :

+ Options

count(filiere)
43

(avec DISTINCT) 4 filières différentes



✓ Affichage des lignes 0 - 0 (total de 1)

```
select count(DISTINCT filiere) from Diplomes
```

Tout afficher | Nombre de lignes :

+ Options

count(DISTINCT filiere)
4

# Requête avec agrégation : autres exemples

Dans la BD Enquete

(à tester vous-même)

- ▶ `SELECT MIN (nb_pages) FROM Livres`
- ▶ `SELECT MAX (nb_pages) FROM Livres`
- ▶ `SELECT SUM (nb_pages) FROM Livres`
- ▶ `SELECT AVG (nb_pages) FROM Livres`



# Faire des statistiques sur une table (1)

la clause **GROUP BY**

**GROUP BY** colonne

groupe les résultats d'une requête qui partagent la même valeur dans la colonne spécifiée

**SELECT** filiere **FROM** Diplomes **GROUP BY** filiere

✓ Affichage des lignes 0 - 3 (total de 4) traitement en 0.0008 sec

```
select filiere from Diplomes group by filiere
```

Profilage

Tout afficher | Nombre de lignes : 25 | Filtrer les li

+ Options

	filiere
<input type="checkbox"/> Éditer Copier Supprimer	Informatique
<input type="checkbox"/> Éditer Copier Supprimer	Sciences de l'education
<input type="checkbox"/> Éditer Copier Supprimer	Sciences du langage
<input type="checkbox"/> Éditer Copier Supprimer	Sciences pour l'ingenieur

# Faire des statistiques sur une table (2)

combinaison de fonctions d'agrégation et **GROUP BY**

## Compter les étudiants

- ▶ nombre de diplômés (43)

```
SELECT COUNT (num_et) FROM Diplomes
```

- ▶ nombre de diplômés *par filière* :

```
SELECT COUNT (num_et) FROM Diplomes GROUP BY filière
```

+ Options

**count(num\_et)**

4

17

13

9

la fonction d'agrégation s'applique à chaque **groupe de résultats**

## Faire des statistiques sur une table (2)

combinaison de fonctions d'agrégation et **GROUP BY**

Compter les coureurs (base : BDD\_L3\_ARMENTA)

- ▶ nombre de participations (16)

```
SELECT COUNT (NumeroCoureur) FROM Participer
```

- ▶ nombre de participations *par coureur* :

```
SELECT COUNT (NumeroCoureur) FROM Participer GROUP BY  
NumeroCoureur
```

count(NumeroCoureur)
1
4
1
1
3
2
1
3

# Rendre les statistiques lisibles

Champs à sélectionner

```
SELECT filiere, COUNT (num_et)  
FROM Diplomes  
GROUP BY filiere
```

filiere	count(num_et)
Informatique	4
Sciences de l'education	17
Sciences du langage	13
Sciences pour l'ingenieur	9

dans une clause **SELECT** comprenant un **GROUP BY**, on ne peut avoir que deux types d'éléments :

- ▶ un élément présent dans la clause **GROUP BY** (ou qui en dépend fonctionnellement)
- ▶ une fonction d'agrégation

# Rendre les statistiques lisibles

Champs à sélectionner

```
SELECT Coureur.NomCoureur, COUNT (Participer.NumeroCoureur)
FROM Participer
JOIN Coureur ON Participer.NumeroCoureur =
Coureur.NumeroCoureur
GROUP BY Coureur.NumeroCoureur
```

NomCoureur	count(Coureur.NumeroCoureur)
Sébastien Reichenbach	1
Peter Sagan	4
Romain Bardet	1
Stefan Küng	1
Egan Bernal	3
Kasper Asgreen	2
Jonathan Castroviejo	1
Geraint Thomas	3

dans une clause **SELECT** comprenant un **GROUP BY**, on ne peut avoir que deux types d'éléments :

- ▶ un élément présent dans la clause **GROUP BY** (ou qui en dépend fonctionnellement)
- ▶ une fonction d'agrégation

# Erreur classique

## Champ mal sélectionné

### Erreur

#### Requête SQL :

```
select Coureur.NomCoureur, Participer.TempsRéalise, count(Coureur.NumeroCoureur) from Participer
join Coureur on Participer.NumeroCoureur=Coureur.NumeroCoureur
group by Coureur.numeroCoureur LIMIT 0, 25
```

#### MySQL a répondu :

```
#1055 - Expression #2 of SELECT list is not in GROUP BY clause and contains nonaggregated column
'BDD_L3_ARMENTA.Participer.TempsRéalise' which is not functionally dependent on columns in GROUP BY clause; this
is incompatible with sql_mode=only_full_group_by
```

le champ Participer.TempsRéalise :

- ▶ ne dépend pas fonctionnellement de Coureur.NumeroCoureur
- ▶ n'est pas une fonction d'agrégation

## GROUP BY sur plusieurs champs

- ▶ nombre de diplômés par filière et *par niveau* :

```
SELECT filiere, niveau, COUNT (num_et)  
FROM Diplomes  
GROUP BY filiere, niveau
```

filiere	niveau	count(num_et)
Informatique	L2	2
Informatique	L3	2
Sciences de l'education	L1	9
Sciences de l'education	L2	3
Sciences de l'education	L3	5
Sciences du langage	L1	8
Sciences du langage	L2	1
Sciences du langage	L3	4
Sciences pour l'ingenieur	L2	5
Sciences pour l'ingenieur	L3	4

## GROUP BY, exemples

(à tester vous-même)

- ▶ durée moyenne des cours par filière :

```
SELECT filiere, AVG (nb_heures)
FROM Programme
GROUP BY filiere
```

- ▶ première date d'obtention de diplôme par filière et par niveau :

```
SELECT filiere, niveau, MIN (date_d)
FROM Diplomes
GROUP BY filiere, niveau
```



Agréger les résultats

Filtrer les résultats après une agrégation

Les alias

Pour finir

## Filtrer les résultats : HAVING

### HAVING condition

restreint les résultats à ceux qui respectent la **condition**

- ▶ la clause 'WHERE condition' restreint avant l'agrégation
- ▶ la clause 'HAVING condition' restreint après l'agrégation

## Filtrer les résultats : HAVING

Si on cherche les livres qui ont été empruntés au moins 3 fois depuis le 1<sup>er</sup> janvier 2012

- ▶ restriction sur la date d'emprunt avec **WHERE**
- ▶ restriction sur le nombre d'emprunts avec **HAVING**

```
SELECT titre, COUNT (date__emp)
FROM Emprunts
WHERE date__emp > "2012-01-01" # restriction sur la date d'emprunt
GROUP BY titre
HAVING COUNT (date__emp) >= 3 # restriction sur le nombre d'emprunts
```

Agréger les résultats

Filtrer les résultats après une agrégation

**Les alias**

Pour finir

# Clarifier les requêtes grâce aux alias

## La clause AS

(optionnel) pour changer le nom de colonnes et de tables

- formater un résultat (alias de colonne) :

```
SELECT num_et AS "Numéro d'étudiant" FROM Etudiants
```

```
select num_et as "Numéro d'étudiant" from Etudiants
```

1 > >> |  Tout afficher | Nombre de lignes :

+ Options



Numéro d'étudiant

<input type="checkbox"/>	Éditer	Copier	Supprimer	1
<input type="checkbox"/>	Éditer	Copier	Supprimer	2
<input type="checkbox"/>	Éditer	Copier	Supprimer	3
<input type="checkbox"/>	Éditer	Copier	Supprimer	4
<input type="checkbox"/>	Éditer	Copier	Supprimer	5

# Clarifier les requêtes grâce aux alias

## La clause AS

(optionnel) pour changer le nom de colonnes et de tables

- ▶ raccourcir les requêtes (alias de table) :

```
SELECT i.num_et, i.annee, i.filiere FROM Inscriptions AS i
```

- ▶ nommer des résultats de fonctions d'agrégation :

```
SELECT num_et, COUNT (annee) AS nb_insc  
FROM Inscriptions  
GROUP BY num_et  
HAVING nb_insc = 1
```

Agréger les résultats

Filtrer les résultats après une agrégation

Les alias

**Pour finir**

CQFR : Ce Qu'il Faut Retenir  
TD



- ▶ maîtrise des fonctions d'agrégation, de **GROUP BY**, de **AS**
- ▶ champs à sélectionner quand on combine fonctions d'agrégation et **GROUP BY**



## Exercice (à faire)

1. Afficher les noms et prénoms des étudiants qui n'ont emprunté aucun livre
2. Afficher les filières dont la somme des heures de cours par semaine est inférieure à 10

## Aide pour l'exercice 1

1. Afficher nom, prénom et dates d'emprunts pour chaque étudiant
2. Grouper les noms et prénoms pour afficher le nombre d'emprunts par étudiant
3. Filtrer les résultats pour ne garder que les étudiants n'ayant emprunté aucun livre