

BASES DE DONNÉES : FONCTIONS D'AGRÉGATION

Les requêtes vues jusqu'ici portaient sur les éléments de la relation ; il est aussi possible d'exprimer des conditions sur des ensembles d'éléments. On dispose pour cela de *fonctions d'agrégation* qui portent sur des sous-ensembles de données (aussi appelés agrégats).

COUNT()	Effectif de chaque sous-ensemble
MAX()	Maximum de l'attribut en argument
MIN()	Minimum de l'attribut en argument
SUM()	Somme de l'attribut en argument
AVG()	Moyenne de l'attribut en argument

I. Un exemple pour la suite

On dispose d'une base de données comportant des informations sur les communes, départements et régions en France. Cette base de données contient trois tables : **villes**, **departements** et **regions**.

La table **regions** contient trois colonnes :

- **id** : un identifiant, sous forme d'entier ;
- **nom** : le nom de la région ;
- **cheflieu** : le code INSEE du chef-lieu de la région.

L'attribut **id** est la clé primaire de cette table.

Extrait :

id	nom	cheflieu
1	Guadeloupe	97105
2	Martinique	97209
3	Guyane	97302
4	La Réunion	97411
6	Mayotte	97608
11	Île-de-France	75056
24	Centre-Val de Loire	45234
27	Bourgogne-Franche-Comté	21231

La table **departements** contient quatre colonnes :

- **id** : un identifiant, sous forme de chaîne de caractères ;
- **nom** : le nom du département ;
- **region** : l'identifiant de la région à laquelle est rattachée le département ;
- **cheflieu** : le code INSEE du chef-lieu du département.

L'attribut **id** est la clé primaire de cette table.

Extrait :

id	nom	region	cheflieu
01	Ain	84	01053
02	Aisne	32	02408
03	Allier	84	03190
04	Alpes-de-Haute-Provence	93	04070
05	Hautes-Alpes	93	05061
06	Alpes-Maritimes	93	06088

La table **villes** contient six colonnes :

- **id** : un identifiant, correspondant au code INSEE de la commune ;
- **dep** : l'identifiant du département dans lequel se trouve la commune ;
- **nom** : le nom de la commune ;
- **codepostal** : le code postal ;
- **population** : le nombre d'habitants de la commune ;
- **surface** : la superficie du territoire communal en km².

L'attribut `id` est la clé primaire de cette table.

Extrait :

id	dep	nom	codepostal	population	surface
01001	01	L'Abergement-Clémenciat	01400	767	15.95
01002	01	L'Abergement-de-Varey	01640	239	9.15
01004	01	Ambérieu-en-Bugey	01500	14022	24.60
01005	01	Ambérieux-en-Dombes	01330	1627	15.92
01006	01	Ambléon	01300	109	5.88
01007	01	Ambronay	01500	2570	33.55
01008	01	Ambrutrix	01500	743	5.22

II. Les fonctions d'agrégation

1) Premiers exemples

— Le nombre de communes en France :

```
SELECT count(*) FROM villes;
```

count(*)
35409

La population communale maximale :

```
SELECT max(population) FROM villes;
```

max(population)
2243833

La population communale minimale :

```
SELECT min(population) FROM villes;
```

min(population)
0

La population communale moyenne :

```
SELECT avg(population) FROM villes;
```

avg(population)
1860.0843

La population totale :

```
SELECT sum(population) FROM villes;
```

sum(population)
65863724

2) GROUP BY

Pour grouper les éléments suivants certains attributs (et non grouper tous les éléments de la relation), on utilise la clause **GROUP BY** suivie des attributs selon lesquels on souhaite grouper les éléments.

Par exemple, pour obtenir à partir de la relation `villes` le nombre de communes et la population totale par département (décrit par son numéro) :

```
SELECT dep, count(*), sum(population)
FROM villes
GROUP BY dep;
```

Extrait :

dep	count(*)	sum(population)
01	408	625820
02	804	539734
03	317	342961
04	198	161241
05	167	139125
06	163	1083312
07	339	322381
08	452	278413
09	331	152554

Si on veut le nom du département plutôt que son identifiant, on utilise une jointure :

```
SELECT d.nom, count(*), sum(population)
FROM villes JOIN departements as d
ON dep = d.id
GROUP BY d.nom;
```

Extrait :

nom	count(*)	sum(population)
Ain	408	625820
Aisne	804	539734
Allier	317	342961
Alpes-de-Haute-Provence	198	161241
Alpes-Maritimes	163	1083312
Ardèche	339	322381

3) ORDER BY

Il est possible d'ordonner les éléments suivant un ou plusieurs attributs en utilisant en fin de requête la clause **ORDER BY** suivie des attributs suivants lesquels trier. Par défaut, le tri se fait dans l'ordre croissant.

— Exemple 1 :

```
SELECT nom, population, surface
FROM villes
ORDER BY population, surface
```

nom	population	surface
Cumières-le-Mort-Homme	0	6.11
Beaumont-en-Verdunois	0	7.87
Louvemont-Côte-du-Poivre	0	8.25
Bezonvaux	0	9.23
Fleury-devant-Douaumont	0	10.27
Haumont-près-Samogneux	0	10.81
Rochefourchat	1	12.74
Leménil-Mitry	3	3.43
Majastres	3	29.85
Trébons-de-Luchon	4	0.83

— Exemple 2 :

```
SELECT nom, population, surface
FROM villes
ORDER BY surface, population
```

nom	population	surface
Castelmoron-d'Albret	51	0.04
Vaudherland	78	0.09
Lannoy	1778	0.18
Mallièvre	257	0.20
La Ferrière-sur-Risle	230	0.24
Sainte-Marie	45	0.28
Bourg-le-Roi	311	0.36
Mont-Louis	179	0.39
Le Catelet	199	0.41
Riverie	303	0.42

On peut réaliser un tri dans l'ordre décroissant en ajoutant le mot clé **DESC** en fin de clause ; le mot clé **ASC** correspondrait à un tri dans l'ordre croissant (comportement par défaut).

Exemple

```
SELECT nom, population
FROM villes
ORDER BY population DESC ;
```

nom	population
Paris	2243833
Marseille	858120
Lyon	506615
Toulouse	466297
Nice	343895
Nantes	298029

Il est aussi possible d'utiliser pour le tri le résultat d'une fonction d'agrégation ; la clause `ORDER BY` doit être située en fin de requête (après `GROUP BY`).

Exemple

```
SELECT r.nom, sum(population)
FROM villes AS v JOIN departements AS d
JOIN regions AS r
ON v.dep = d.id AND d.region = r.id
GROUP BY r.id
ORDER BY sum(population) DESC
```

nom	sum(population)
Ile-de-France	12046136
Auvergne-Rhône-Alpes	7730962
Hauts-de-France	6003965
Nouvelle-Aquitaine	5856698
Occitanie	5725032
Grand Est	5548001
Provence-Alpes-Côte d'Azur	4982333

4) HAVING

On peut souhaiter réaliser une sélection sur un critère dépendant du résultat d'une fonction d'agrégation. Il n'est pas possible d'ajouter ce critère dans la clause `WHERE`, qui ne peut utiliser que des attributs de la relation. On ajoute alors une clause `HAVING` (suivie de la condition) immédiatement après la clause `GROUP BY`.

Exemple

```
SELECT d.id, d.nom, sum(population)
FROM villes JOIN departements AS d
ON villes.dep = d.id
GROUP BY d.id
HAVING sum(population) > 1500000
ORDER BY d.id ;
```

id	nom	sum(population)
13	Bouches-du-Rhône	2006069
33	Gironde	1524347
59	Nord	2603472
69	Rhône	1796624
75	Paris	2243833
92	Hauts-de-Seine	1597770
93	Seine-Saint-Denis	1571028

Exercice

On dispose de trois tables dont voici des extraits :

eleves			
id	nom	prenom	classe
1	ALCAZAR	Peggy	MPSI
2	HADDOCK	Archibald	PCSI
3	TOURNESOL	Tryphon	PCSI

colleurs		
id	nom	prenom
1	DUPOND	Martin
2	DUPONT	Jacques
3	MARTIN	Claudie

colles			
semaine	eleve	colleur	note
1	2	3	14
2	3	3	9
2	1	5	12

Traduire en langage SQL les requêtes suivantes :

- Calculer le nombre total de notes.
- Calculer la moyenne de toutes les notes.
- Calculer la moyenne des colles par semaine.
- Calculer la moyenne de chaque élève ; trier par moyenne croissante.
- Déterminer la note maximale obtenu par l'élève TOURNESOL.
- Déterminer la note minimale mise par le colleur DUPONT.
- Déterminer le nombre de colles où l'élève ALCAZAR a été interrogée par le colleur MARTIN.
- Déterminer les noms des colleurs n'ayant jamais mis de note en dessous de 5.
- (*) Déterminer l(es) élève(s) ayant obtenu la meilleure note en semaine 5.