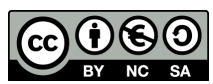
BDBIO - Manipulation relationnelle - SQL avancé

Fabien Duchateau

fabien.duchateau [at] univ-lyon1.fr

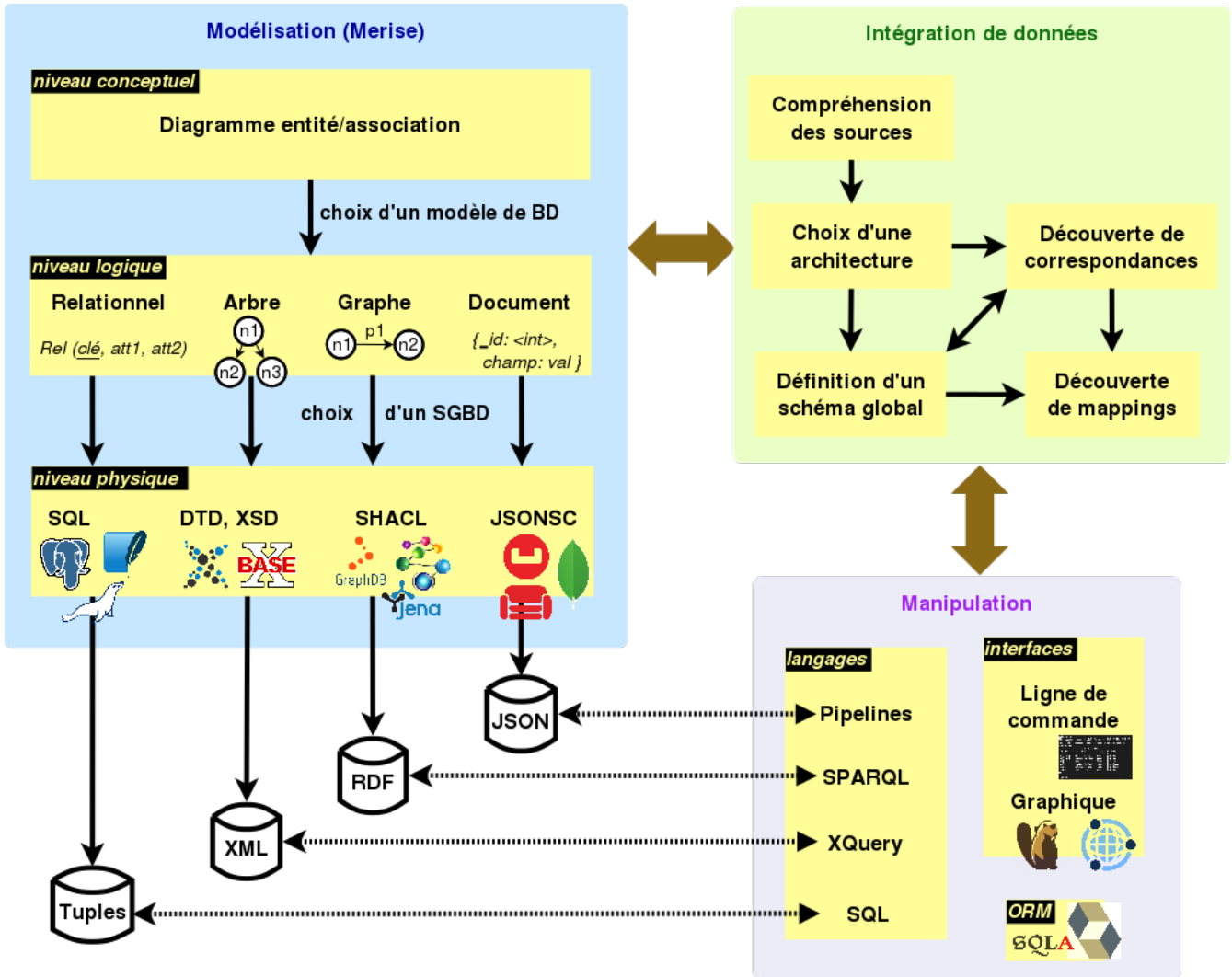
Université Claude Bernard Lyon 1

2024 - 2025



https://perso.liris.cnrs.fr/fabien.duchateau/BDBIO/

Aperçu des notions de BDBIO



Ces diapositives utilisent le genre féminin (e.g., chercheuse, développeuses) plutôt que l'écriture inclusive (moins accessible, moins concise, et pas totalement inclusive)

Rappels

- Syntaxe d'une requête SQL (crochets = option) :

  SELECT att_1 [, att_2 [**AS** att'_2], ...]
  FROM nom_table_1 [, nom_table_2 [alias]]
  [**WHERE** *condition*]
  [**ORDER BY** att_i [, att_j, ...]]
  [**LIMIT** n] **;**

- Différents types de jointure (dans la clause FROM)

- Des fonctions pour manipuler les chaînes de caractères, dates, numériques, etc.

Motivation

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Comment retourner ...

- Les universités qui ont un effectif supérieur à celui de *UCB* ?
- La moyenne des élèves qui ont candidaté en biologie ?
- Les universités qui ont plus de trois candidatures ?

Motivation

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Comment retourner ...

- Les universités qui ont un effectif supérieur à celui de *UCB* ?
- La moyenne des élèves qui ont candidaté en biologie ?
- Les universités qui ont plus de trois candidatures ?

Notion de sous-requêtes, de groupes et de filtres sur les groupes

Plan

Sous-requêtes

Clause GROUP BY

Fonctions liées au GROUP BY

Clause HAVING

Généralités

Une sous-requête est une requête incluse dans une autre requête (principale), i.e., la requête principale utilise le résultat de la sous-requête

- Augmentation de la puissance d'expression du langage
- Les sous-requêtes sont utilisables dans les clauses :
  - SELECT (peu fréquent)
  - FROM (à condition de renommer le résultat)
  - WHERE et HAVING

---

http://www.postgresql.org/docs/current/functions-subquery.html

Généralités

Une sous-requête est une requête incluse dans une autre requête (principale), i.e., la requête principale utilise le résultat de la sous-requête

- Augmentation de la puissance d'expression du langage
- Les sous-requêtes sont utilisables dans les clauses :
  - SELECT (peu fréquent)
  - FROM (à condition de renommer le résultat)
  - WHERE et HAVING

Le résultat des (sous-)requêtes est variable en termes de n-uplets (et d'attributs) : sous-requêtes **scalaires** et **multi-lignes**

---

http://www.postgresql.org/docs/current/functions-subquery.html

Sous-requête scalaire

- Retourne une seule valeur (i.e., un n-uplet et un attribut)
- S'utilise (généralement) là où une requête attend un nom d'attribut ou une constante
- Ci après, illustration d'une sous-requête scalaire dans deux clauses (WHERE et FROM)

Exemple de requête scalaire :

SELECT nomE
FROM *E*
WHERE idE = 123 ;

nomE
Ana

Exemple de sous-requête scalaire dans la clause WHERE

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table ÉLÈVE

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table UNIVERSITÉ

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

Les universités qui sont dans la même ville que *UCB*

```
SELECT *
FROM U
WHERE ville = (SELECT ville
               FROM U
               WHERE nomU = 'UCB');
```

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000

Exemple de sous-requête scalaire dans la clause FROM

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Les universités qui sont dans la même ville que *UCB*

```
SELECT *
FROM U, (SELECT ville
         FROM U
         WHERE nomU = 'UCB') VilleUCB
WHERE U.ville = VilleUCB.ville;
```

nomU	ville	effectif	ville
INSA	Lyon	36000	Lyon
UCB	Lyon	15000	Lyon

Sous-requête multi-lignes

- Retourne plusieurs tuples / lignes
- Nécessite des opérateurs (par exemple pour comparer une valeur avec toutes celles retournées par la sous-requête)

Exemple de requête multi-lignes :

SELECT nomU
FROM *U*
WHERE ville = 'Lyon' ;

nomU
INSA
UCB

Opérateurs pour sous-requêtes multi-lignes

- *att* Θ **ANY** (*sous_requete*)
  avec Θ parmi $\{=, <, >, <=, >=, !=\}$
  - vrai s'il existe un *b* parmi les lignes renvoyées par *sous_requete* tel que *att* Θ *b* soit vrai

- *att* [**NOT**] **IN** (*sous_requete*)
  - vrai si la valeur de *att* apparaît dans le résultat de *sous_requete* (ou n'apparait pas dans le cas du **NOT IN**)
  - **IN** est équivalent à $=$ **ANY**
  - **NOT IN** est équivalent à $!=$ **ANY**

---

http://www.postgresql.org/docs/current/functions-subquery.html#FUNCTIONS-SUBQUERY-ANY-SOME

http://www.postgresql.org/docs/current/functions-subquery.html#FUNCTIONS-SUBQUERY-IN

Opérateurs pour sous-requêtes multi-lignes (2)

- *att* Θ **ALL** (*sous_requete*)
  avec Θ parmi $\{=, <, >, <=, >=, !=\}$
  - vrai si pour toutes les lignes *b* renvoyées par *sous_requete*, *att* Θ *b* est vrai

- [**NOT**] **EXISTS** (*sous_requete*)
  - vrai si *sous_requete* retourne au moins un tuple résultat (ou ne retourne pas de tuple dans le cas du **NOT EXISTS**)
  - s'utilise principalement avec sous-requêtes corrélées

---

http://www.postgresql.org/docs/current/functions-subquery.html#FUNCTIONS-SUBQUERY-ALL

http://www.postgresql.org/docs/current/functions-subquery.html#FUNCTIONS-SUBQUERY-EXISTS

Exemple de sous-requêtes multi-lignes

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table ÉLÈVE

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table UNIVERSITÉ

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

Les élèves qui ont candidaté à *Lyon*

```
SELECT DISTINCT C.idE, nomE
FROM C JOIN E USING(idE)
WHERE nomU IN (SELECT nomU
               FROM U
               WHERE ville = 'Lyon');
```

idE	nomE
123	Ana
234	Bob
678	Farid
765	Joana
876	Irene
898	Hector

Exemple de sous-requêtes multi-lignes (2)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

La ou les universités qui ont le plus grand effectif

SELECT *
FROM *U*
WHERE effectif $\geq$ **ALL** (**SELECT** effectif
FROM *U*) ;

nomU	ville	effectif
INSA	Lyon	36000

Plan

Sous-requêtes

Clause GROUP BY

Fonctions liées au GROUP BY

Clause HAVING

Syntaxe

SELECT att_1 [, att_2 [**AS** att'_2], ...]
FROM nom_table_1 [, nom_table_2 [alias]]
[**WHERE** *condition*]
[**GROUP BY** att_k [, att_l, ...]]
[**ORDER BY** att_i [, att_j, ...]]
[**LIMIT** n] ;

Le GROUP BY, exécuté après le WHERE, indique de procéder à une répartition du résultat en groupes de n-uplets :

- Deux n-uplets sont dans un même groupe s'ils ont les mêmes valeurs pour chaque attribut de regroupement att_k, att_l, ...

http://www.postgresql.org/docs/current/queries-table-expressions.html#QUERIES-GROUP

Exemple de regroupement

SELECT nomE
FROM *E*
WHERE idE < 700 **;**

nomE
Ana
Bob
Chloe
Damien
Chloe
Eleonore
Ana
Farid

Exemple de regroupement

SELECT nomE
FROM *E*
WHERE idE < 700 **;**

nomE
Ana
Bob
Chloe
Damien
Chloe
Eleonore
Ana
Farid

Exemple de regroupement

```
SELECT nomE
FROM E
WHERE idE < 700;
```

nomE
Ana
Bob
Chloe
Damien
Chloe
Eleonore
Ana
Farid

```
SELECT nomE
FROM E
WHERE idE < 700
GROUP BY nomE;
```

Exemple de regroupement

```
SELECT nomE
FROM E
WHERE idE < 700;
```

nomE
Ana
Bob
Chloe
Damien
Chloe
Eleonore
Ana
Farid

```
SELECT nomE
FROM E
WHERE idE < 700
GROUP BY nomE;
```

nomE
Ana
Chloe
Damien
Bob
Eleonore
Farid

Conséquences du regroupement

- La requête ne renvoie **qu'un seul n-uplet par groupe**

- Le SELECT et le ORDER BY ne peuvent utiliser que des attributs présents dans le GROUP BY
  - dans un groupe, un attribut de regroupement a la même valeur pour tous les n-uplets, donc utilisable directement
  - les autres attributs ont des valeurs diverses, donc non utilisables directement (quelle valeur utiliser ?)

On peut quand même utiliser les attributs qui ne font pas partie des attributs de regroupement via des fonctions !

Plus exactement, le comportement est indéfini (erreur, résultat incorrect, etc.)

Plan

Sous-requêtes

Clause GROUP BY

Fonctions liées au GROUP BY

Clause HAVING

Fonctions utilisées avec les regroupements

Fonctions agissant sur un ensemble de valeurs atomiques :

- Compter un nombre d'élèves
- Calculer la moyenne des effectifs
- ...

Comment les utiliser ?

- Si aucun GROUP BY n'est spécifié dans la requête, la fonction s'applique à toutes les lignes. La requête retourne alors un seul n-uplet
- Si un GROUP BY est spécifié dans la requête, la fonction s'applique pour chaque groupe. La requête retourne alors un n-uplet par groupe

---

http://www.postgresql.org/docs/current/tutorial-agg.html

Fonctions utilisées avec les regroupements (2)

- Où utiliser ces fonctions liées au GROUP BY ?
  - Dans le SELECT et dans le ORDER BY
  - **Pas** dans le WHERE (qui a lieu **avant** regroupement)
- L'intérêt de ces fonctions est de permettre l'utilisation des attributs qui ne sont pas des attributs de regroupements
- Le mot clé DISTINCT peut être placé dans la fonction d'agrégation (avant l'attribut) pour ne prendre en compte que les valeurs distinctes

Une dizaine de fonctions, dont COUNT, MIN/MAX, AVG, SUM

http://www.postgresql.org/docs/current/functions-aggregate.html

Fonction COUNT

COUNT(att) : le nombre de valeurs de l'attribut *att*

COUNT(**DISTINCT** att) : le nombre de valeurs distinctes de l'attribut *att*

COUNT(*) : le nombre de n-uplets

- Les valeurs non définies (NULL) ne sont pas comptées

Exemple de COUNT sans regroupement

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table ÉLÈVE

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table UNIVERSITÉ

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

Le nombre d'élèves

SELECT COUNT(*)
FROM *E* ;

ou

SELECT COUNT(idE)
FROM *E* ;

count
12

Exemple de COUNT avec DISTINCT

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table ÉLÈVE

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table UNIVERSITÉ

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

Le nombre de noms d'élèves distincts

SELECT COUNT(DISTINCT nomE)
FROM *E* **;**

count
10

Déroulement pas à pas d'une requête avec regroupement

Le nombre de candidatures par université, avec tri par nom :

SELECT nomU, **COUNT**(*) **FROM** *C*
GROUP BY nomU **ORDER BY** nomU **;**

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

Déroulement pas à pas d'une requête avec regroupement

Le nombre de candidatures par université, avec tri par nom :

SELECT nomU, **COUNT**(*) **FROM** *C*
GROUP BY nomU **ORDER BY** nomU **;**

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Déroulement pas à pas d'une requête avec regroupement

Le nombre de candidatures par université, avec tri par nom :

SELECT nomU, **COUNT**(*) **FROM** *C*
GROUP BY nomU **ORDER BY** nomU **;**

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

$\Rightarrow$

nomU	count
UJM	6
UCB	6
UJF	4
INSA	3

Déroulement pas à pas d'une requête avec regroupement

Le nombre de candidatures par université, avec tri par nom :

SELECT nomU, **COUNT**(*) **FROM** *C*
GROUP BY nomU **ORDER BY** nomU ;

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

$\Rightarrow$

nomU	count
UJM	6
UCB	6
UJF	4
INSA	3

$\Rightarrow$

nomU	count
INSA	3
UCB	6
UJF	4
UJM	6

Exemple de regroupement sur plusieurs attributs

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table ÉLÈVE

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table UNIVERSITÉ

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

Nombre de candidatures par université puis par département pour les universités *INSA* et *UCB*

SELECT nomU, département, **COUNT**(*)
FROM *C*
WHERE nomU **IN** ('INSA', 'UCB')
GROUP BY nomU, département ;

nomU	département	count
INSA	biologie	1
INSA	informatique	2
UCB	électronique	1
UCB	histoire	2
UCB	informatique	3

Piège à éviter

Quelle requête correspond au "*nombre de candidatures pour chaque élève*" ?

```
SELECT nomE, COUNT(*)
FROM E INNER JOIN C
ON E.idE = C.idE
GROUP BY nomE;
```

```
SELECT E.idE, nomE, COUNT(*)
FROM E INNER JOIN C
ON E.idE = C.idE
GROUP BY E.idE, nomE;
```

Piège à éviter

Quelle requête correspond au "*nombre de candidatures pour chaque élève*" ?

SELECT nomE, **COUNT**(*)
FROM *E* **INNER JOIN** *C*
ON *E*.idE = *C*.idE
GROUP BY nomE ;

⇓

nomE	count
Joana	3
Ana	4
Chloe	5
Irene	3
Hector	2
Bob	1
Farid	1

SELECT *E*.idE, nomE, **COUNT**(*)
FROM *E* **INNER JOIN** *C*
ON *E*.idE = *C*.idE
GROUP BY *E*.idE, nomE ;

⇓

idE	nomE	count
765	Joana	3
123	Ana	4
345	Chloe	4
876	Irene	3
543	Chloe	1
898	Hector	2
234	Bob	1
678	Farid	1

Fonctions MIN et MAX

MIN(att) : la valeur minimale de l'attribut *att*

MAX(att) : la valeur maximale de l'attribut *att*

- Le mot clé DISTINCT n'a pas d'effet avec ces fonctions

http://www.geluck.com/

Exemple de MIN/MAX sans regroupement

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table ÉLÈVE

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table UNIVERSITÉ

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

La moyenne la plus élevée

SELECT MAX(moyenneLycee) **AS** moyenneMax
FROM *E* **;**

moyenneMax
19.5

Fonctions AVG et SUM

AVG(att) : la moyenne des valeurs de l'attribut *att*

SUM(att) : la somme des valeurs de l'attribut *att*

http://www.geluck.com/

Exemple de AVG avec regroupement

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table ÉLÈVE

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table UNIVERSITÉ

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

La moyenne des candidat.e.s par université (avec arrondi)

SELECT nomU, **ROUND**(**AVG**(moyenneLycee))
FROM *E* **INNER JOIN** *C*
ON *E*.idE = *C*.idE
GROUP BY nomU ;

nomU	round
UJM	17
UCB	18
UJF	18
INSA	19

Un moment de réflexion

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Que doit-on changer dans la requête précédente pour obtenir la ou les universités dont la moyenne de ses candidat.e.s est la plus faible ?

Un moment de réflexion

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Que doit-on changer dans la requête précédente pour obtenir la ou les universités dont la moyenne de ses candidat.e.s est la plus faible ?

⇒ Créer une sous-requête à partir de la requête précédente, et récupérer la moyenne minimale dans la requête principale

Autres fonctions SQL

- **STD**(att) ou **STDDEV_POP**(att) : l'écart-type de *att*
- **VAR_POP**(att) ou **VARIANCE**(att) : la variance de *att*
- **STRING_AGG**(att), **RANGE_AGG**(att) : une concaténation des valeurs de *att* dans une chaîne de caractères ou une liste de nombres
- ...

https://www.postgresql.org/docs/16/functions-aggregate.html

Plan

Sous-requêtes

Clause GROUP BY

Fonctions liées au GROUP BY

Clause HAVING

Syntaxe

SELECT att_1 [, att_2 [**AS** att'_2], ...]
FROM nom_table_1 [, nom_table_2 [alias]]
[**WHERE** *condition*]
[**GROUP BY** att_k [, att_l, ...]]
[**HAVING** *condition_groupe*]
[**ORDER BY** att_i [, att_j, ...]]
[**LIMIT** n] ;

La clause HAVING sélectionne les groupes qui satisfont la condition *condition_groupe*

- Une clause GROUP BY est requise
- Exécutée entre le GROUP BY et le ORDER BY

Condition du HAVING

- Le WHERE ne porte que sur les n-uplets individuels, **avant regroupement**

- La condition du HAVING porte sur les groupes et pas sur les n-uplets individuels :
  - utilisation directe des attributs de regroupement
  - utilisation des autres attributs à travers les fonctions liées au regroupement
  - évaluation booléenne (comme la condition du WHERE)

---

http://www.postgresql.org/docs/current/queries-table-expressions.html#QUERIES-GROUP

Déroulement pas à pas d'une requête avec HAVING

Les départements d'université avec au moins 2 candidatures :

SELECT nomU, département, **COUNT**(*) **AS** nbC **FROM** *C*
GROUP BY nomU, département **HAVING COUNT**(*) $\geq$ 2**;**

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

Déroulement pas à pas d'une requête avec HAVING

Les départements d'université avec au moins 2 candidatures :

SELECT nomU, département, **COUNT**(*) **AS** nbC **FROM** *C*
GROUP BY nomU, département **HAVING COUNT**(*) $\geq$ 2**;**

nomU	département
INSA	informatique
UCB	électronique
UCB	informatique
UJM	électronique
INSA	biologie
UJF	bioinformatique
UJM	bioinformatique
UJM	électronique
UJM	informatique
UJF	informatique
UCB	histoire
UCB	histoire
UJM	histoire
UJM	psychologie
UCB	informatique
UJF	biologie
UJF	biologie marine
INSA	informatique
UCB	informatique

Déroulement pas à pas d'une requête avec HAVING

Les départements d'université avec au moins 2 candidatures :

SELECT nomU, département, **COUNT**(*) **AS** nbC **FROM** *C*
GROUP BY nomU, département **HAVING COUNT**(*) ≥ 2 **;**

nomU	département
INSA	informatique
UCB	électronique
UCB	informatique
UJM	électronique
INSA	biologie
UJF	bioinformatique
UJM	bioinformatique
UJM	électronique
UJM	informatique
UJF	informatique
UCB	histoire
UCB	histoire
UJM	histoire
UJM	psychologie
UCB	informatique
UJF	biologie
UJF	biologie marine
INSA	informatique
UCB	informatique

nomU	département	nbC
INSA	informatique	2
UCB	électronique	1
UCB	informatique	3
UJM	électronique	2
INSA	biologie	1
UJF	bioinformatique	1
UJM	bioinformatique	1
UJM	informatique	1
UJF	informatique	1
UCB	histoire	2
UJM	histoire	1
UJM	psychologie	1
UJF	biologie	1
UJF	biologie marine	1

Déroulement pas à pas d'une requête avec HAVING

Les départements d'université avec au moins 2 candidatures :

SELECT nomU, département, **COUNT**(*) **AS** nbC **FROM** *C*
GROUP BY nomU, département **HAVING COUNT**(*) ≥ 2 **;**

nomU	département
INSA	informatique
UCB	électronique
UCB	informatique
UJM	électronique
INSA	biologie
UJF	bioinformatique
UJM	bioinformatique
UJM	électronique
UJM	informatique
UJF	informatique
UCB	histoire
UCB	histoire
UJM	histoire
UJM	psychologie
UCB	informatique
UJF	biologie
UJF	biologie marine
INSA	informatique
UCB	informatique

nomU	département	nbC
INSA	informatique	2
UCB	électronique	1
UCB	informatique	3
UJM	électronique	2
INSA	biologie	1
UJF	bioinformatique	1
UJM	bioinformatique	1
UJM	informatique	1
UJF	informatique	1
UCB	histoire	2
UJM	histoire	1
UJM	psychologie	1
UJF	biologie	1
UJF	biologie marine	1

nomU	département	nbC
INSA	informatique	2
UCB	histoire	2
UCB	informatique	3
UJM	électronique	2

Exemple de HAVING

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table ÉLÈVE

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table UNIVERSITÉ

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

Élèves ayant candidaté dans une université avec au moins 5 disciplines

SELECT *E*.idE, nomE, **COUNT**(*) **AS** nbC

FROM *E* **JOIN** *C* **USING**(idE)

WHERE nomU **IN**

(**SELECT** nomU **FROM** *C* **GROUP BY** nomU

HAVING COUNT(**DISTINCT** département) $\geq$ 5)

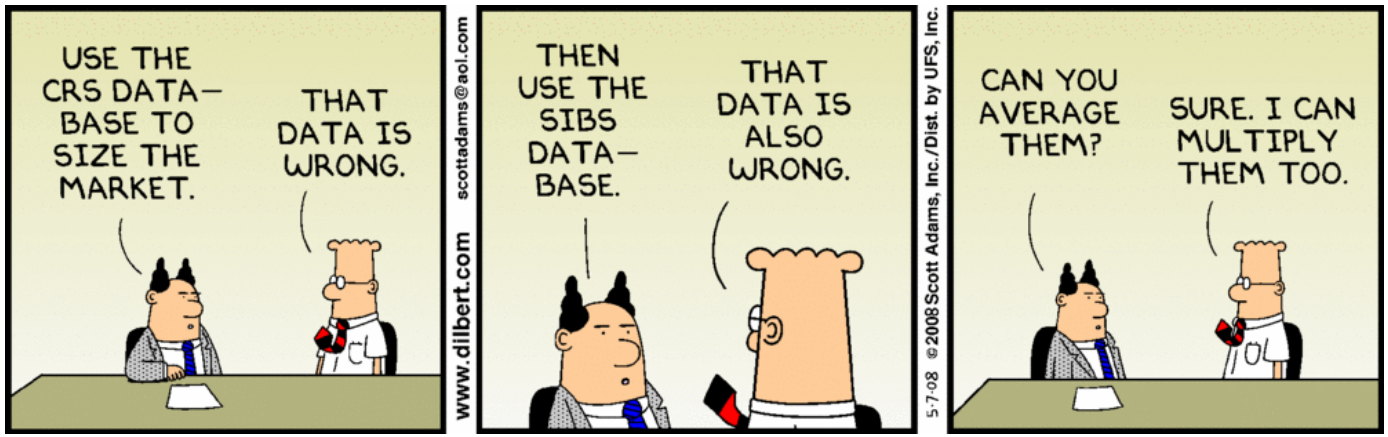
GROUP BY *E*.idE, nomE ;

idE	nomE	nbC
123	Ana	1
765	Joana	2
345	Chloe	3

En résumé

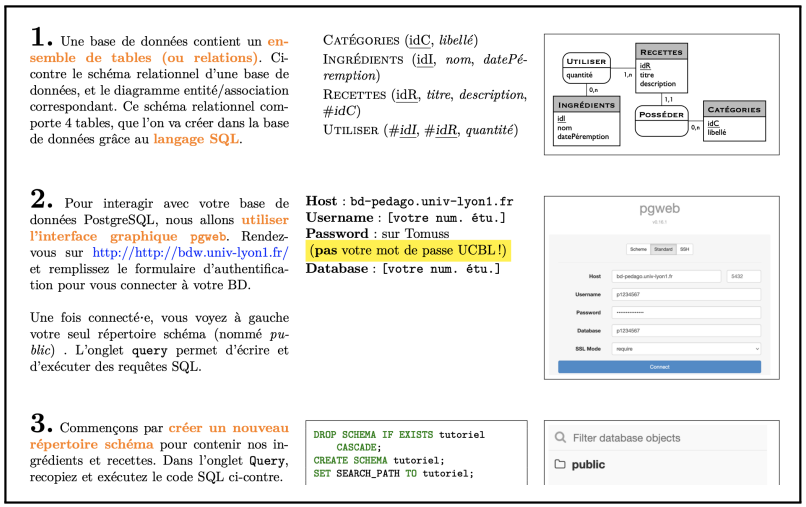
- Imbrication de sous-requêtes
  - Bien réfléchir au résultat retourné par une sous-requête (scalaire, multi-lignes, et multi-colonnes)
  - Utiliser un opérateur approprié pour intégrer la sous-requête

- Clauses GROUP BY et HAVING pour les groupes
  - Fonctions liées au regroupement (moyenne, somme, etc.)



Travail personnel

Suivre les tutoriels SQL (disponibles sur la page de l'UE) pour préparer le TD et TP (~15 minutes par tutoriel)



1. Une base de données contient un ensemble de tables (ou relations). Ci-contre le schéma relationnel d'une base de données, et le diagramme entité/association correspondant. Ce schéma relationnel comporte 4 tables, que l'on va créer dans la base de données grâce au langage SQL.

CATÉGORIES (idC, *libellé*)
INGRÉDIENTS (idI, *nom*, *datePéremption*)
RECETTES (idR, *titre*, *description*, *#idC*)
UTILISER (*#idI*, *#idR*, *quantité*)

UTILISER — quantité — 1,n — 0,n — RECETTES (idR, titre, description) — 1,1 — POSSÉDER — 0,n — CATÉGORIES (idC, libellé)
INGRÉDIENTS (idI, nom, datePéremption) — 0,n — POSSÉDER

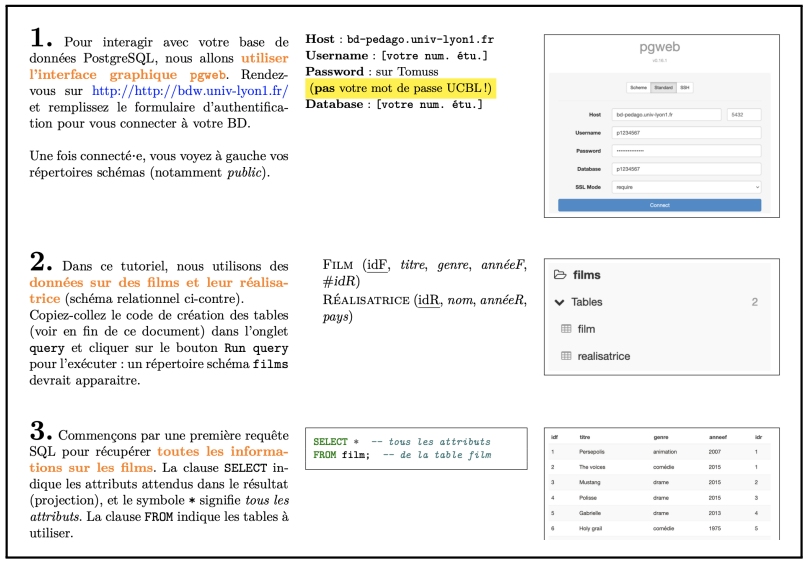
2. Pour interagir avec votre base de données PostgreSQL, nous allons utiliser l'interface graphique `pgweb`. Rendez-vous sur http://http://bdw.univ-lyon1.fr/ et remplissez le formulaire d'authentification pour vous connecter à votre BD.

Une fois connecté·e, vous voyez à gauche votre seul répertoire schéma (nommé *public*) . L'onglet `query` permet d'écrire et d'exécuter des requêtes SQL.

Host : `bd-pedago.univ-lyon1.fr`
Username : `[votre num. étu.]`
Password : sur Tomuss
(**pas** votre mot de passe UCBL!)
Database : `[votre num. étu.]`

3. Commençons par créer un nouveau répertoire schéma pour contenir nos ingrédients et recettes. Dans l'onglet `Query`, recopiez et exécutez le code SQL ci-contre.

```
DROP SCHEMA IF EXISTS tutoriel
    CASCADE;
CREATE SCHEMA tutoriel;
SET SEARCH_PATH TO tutoriel;
```

1. Pour interagir avec votre base de données PostgreSQL, nous allons utiliser l'interface graphique `pgweb`. Rendez-vous sur http://http://bdw.univ-lyon1.fr/ et remplissez le formulaire d'authentification pour vous connecter à votre BD.

Une fois connecté·e, vous voyez à gauche vos répertoires schémas (notamment *public*).

Host : `bd-pedago.univ-lyon1.fr`
Username : `[votre num. étu.]`
Password : sur Tomuss
(**pas** votre mot de passe UCBL!)
Database : `[votre num. étu.]`

2. Dans ce tutoriel, nous utilisons des données sur des films et leur réalisatrice (schéma relationnel ci-contre).
Copiez-collez le code de création des tables (voir en fin de ce document) dans l'onglet `query` et cliquer sur le bouton `Run query` pour l'exécuter : un répertoire schéma `films` devrait apparaitre.

FILM (idF, *titre*, *genre*, *annéeF*, *#idR*)
RÉALISATRICE (idR, *nom*, *annéeR*, *pays*)

3. Commençons par une première requête SQL pour récupérer toutes les informations sur les films. La clause `SELECT` indique les attributs attendus dans le résultat (projection), et le symbole `*` signifie *tous les attributs*. La clause `FROM` indique les tables à utiliser.

```
SELECT *  -- tous les attributs
FROM film;  -- de la table film
```


Tutoriel SQL en tant que LDD (gauche) et LMD (droite)