# Requêtes SQL

Les exemples et exercices donnés ci-dessous sont, sauf mention contraire, disponibles directement dans un notebook Capytale.

# 1. Projections

# **i** Projection

L'opération de **projection** consite à ne récupérer que les champs (= les colonnes) d'une table donnée. En SQL, on l'obtiens par l'instruction :

```
SELECT
colonne1, colonne2,...
FROM
nom_table;
```

# **Exemple 1 : Projection**

Pour récupérer les colonnes titre et isbn de la table livre :

```
SELECT
titre, isbn
FROM
livre;
```

Pour récupérer l'intégralité des colonnes, on peut utiliser l'opérateur joker \* :

```
SELECT

*
FROM
auteur;
```

#### Renommer les colonnes

Il est possible dans une opération de projection de renommer les colonnes obtenues, grâce à l'opérateur AS:

```
SELECT
titre, isbn, annee AS annee_publication
FROM
livre;
```

# 2. Sélections



L'opération de **sélection** consiste à interroger une base de données pour ne récupérer que les lignes d'une table correspondant à une ou des conditions spécifiées.

En SQL, on rajoute la **clause** WHERE suivie des conditions exprimées sous la forme d'une **expression booléenne**, utilisant les mots clés AND et OR par exemple :

```
SELECT
colonne1, colonne2,...
FROM
nom_table
WHERE
conditions;
```

### Exemple 2 : Sélection

• Sélection avec condition unique:

```
SELECT
    titre
FROM
    livre
WHERE
    annee >= 2020;
```

Sélection avec conditions multiples :

```
SELECT
   titre
FROM
   livre
WHERE
   annee >= 1970 AND
   annee <= 2000 AND
   editeur='Dargaud';</pre>
```

### Requête sur les chaînes de caractères

Si on veut chercher tous les livres dont le titre contient la chaîne Astérix, il faudra utiliser une clause comme la suivante :

```
SELECT
titre
FROM
livre
WHERE
titre LIKE '%Astérix%';
```

La chaîne de caractères '%Astérix%' s'appelle un motif. L'opération s LIKE m renverra True si la chaîne de caractère s correspond au motif m. Le caractère % est un joker qui peut-être substitué par n'importe quelle chaîne. Il existe aussi l'opérateur \_ (underscore) qui lui représente n'importe quel caractère. Ainsi, pour chercher tous les auteurs dont le nom commence par F, se termine par R et fait 6 caractères de long :

```
SELECT

nom, prenom
FROM

auteur
WHERE

nom LIKE 'F___R';
```

# 3. Fonctions d'aggrégations

Il existe un certain nombre de fonctions permettant d'effectuer des opérations sur des colonnes. Ces fonctions s'appellent *fonctions* d'aggrégations, et renvoie un résultat sous al forme d'une table d'une ligne et d'une colonne. Voici les plus utiles :

### 3.1. Fonction COUNT

# **i** Compter des lignes

La fonction SQL COUNT permet de compter le nombre de lignes que possède une table, éventuellement aporès sélection. Sa syntaxe est :

```
SELECT
COUNT(colonne)
FROM
table
WHERE conditions;
```

### **Exemples**

• COmpter le nombre de lignes dans la table auteur :

```
SELECT
COUNT(*)
FROM
auteur;
```

• Compter le nombre de titres contenant le chaîne Astérix

```
SELECT
    count(titre)
FROM
    livre
WHERE titre LIKE '%Astérix%';
```

### 🛕 Un piège

L'ordre SQL suivant

```
SELECT
  count(titre), isbn
FROM
  livre
WHERE titre LIKE '%Astérix%';
```

renvoie une table avec une ligne et deux colonnes :

L'isbn renvoyé ne correspond qu'au premier titre trouvé contenant la chaîne Astérix :

```
SELECT
titre, isbn
FROM
livre
WHERE isbn = '978-2864972662';
```



Avec la fonction COUNT, les titres des colonnes renvoyés ne sont pas forcémùent parlant. Il est possible de les changer en leur fournisdsant un **alias** par l'intermédiaire de AS:

```
SELECT
    count(titre) AS nombre_asterix
FROM
    livre
WHERE titre LIKE '%Astérix%';
```

## 3.2. Fonctions numériques

Les fonctions suivantes ne peuvent s'appliquer que sur des colonnes dont le type est numérique :

- SUM : effectue la somme de toutes les valeurs de la colonne sélectionnée correspondant au conditions données
- AVG (average): effectue la moyenne de toutes les valeurs de la colonne sélectionnée correspondant au conditions données.

```
SELECT SUM(annee) as somme FROM livre ;
SELECT AVG(annee) as moyenne FROM livre ;
```

#### 3.3. Fonctions MIN et MAX

Ces deux fonctions s'appliquent sur n'importe quel type, l'ordre sur les chaînes de caractères étant l'ordre lexicographique. :

```
SELECT MIN(nom) FROM auteur ;
SELECT MAX(nom) FROM auteur ;
```

# 4. Tri et suppression des doublons

### 4.1. Tri des colonnes

Les résultats d'une requête SQL sont en général fournis dans l'ordre dans lequel ils sont trouvés. Il est cependant possible d'**ordonner** les colonnes grâce à la clause ORDER BY et les mots clés ASC (ascending) et DESC (descending):

• par ordre croissant :

```
SELECT
titre

FROM
livre

WHERE annee >=1990
ORDER BY titre ASC;
```

• par ordre décroissant :

```
SELECT

nom

FROM

auteur

ORDER BY nom DESC;
```

#### 4.2. Elimination des doublons

Effectuons la requête suivante :

```
SELECT prenom FROM auteur WHERE prenom LIKE 'M%';
```



Le résultat est la table suivante :

Nous constatons la présence de 3 prénoms Michel dans la table résultat. Il est possible d'éliminer de tels doublons dans une table en utilisant la clause DISTINCT :



On récupère alors en résultat la table suivante :



### Attention

Attention toutefois! Une requête telle que la suivante n'élimineras pas les doublons de prénom:

```
SELECT DISTINCT prenom, nom FROM auteur WHERE prenom LIKE 'M%';
```

En effet la clause DISTINCT élimine les lignes exactement identiques. Ici les couples (prenom, nom) sont bien tous différents.

# 5. Application



### Exercice

Effectuer la première partie ainsi que les requêtes sans jointures du notebokk jeux olympiques (merci M. Leleu).