Memento SQL

► **Syntaxe générale d'une requête SQL** :

```
SELECT attributs, fonctions d'agrégation (AS...)
FROM table (avec jointure éventuelle ... JOIN ... ON ... = ...)
WHERE conditions
GROUP BY attributs
HAVING conditions
ORDER BY attributs (... ASC ou DESC)
```

 La clause `WHERE` opère une sélection sur la table.
La clause `HAVING` opère une sélection sur les groupes. Elle n'a pas de sens sans `GROUP BY`.

► **Fonctions d'agrégation** :

```
MIN(a) (minimum)    MAX(a) (maximum)    AVG(a) (moyenne)    SUM(a) (somme)

COUNT(a)    COUNT(DISTINCT a) (élimine les doublons)    COUNT(*) (compte les lignes)
```

► **Projection** :

On peut récupérer des attributs et des fonctions d'agrégation :

```
SELECT nom, 10*AVG(prix), MAX(prix)

SELECT AVG(note) AS moy (ceci est un renommage)

SELECT DISTINCT nom (pour enlever les doublons)
```

 La projection se fait dans le `SELECT` ! Ne pas confondre avec la sélection qui se fait dans le `WHERE`.

► **Sélection** :

```
WHERE (prix > 100 AND stock < 50) OR (num_prod = 12)
```

La condition peut être formée avec

```
AND    OR    NOT    <    <=    >    >=    =    <> (ou !=)
```

► **Jointure** :

```
SELECT ...
FROM T1 JOIN T2
    ON T1.a1 = T2.a2
```

Jointure sur deux tables

```
SELECT ...
FROM T1 JOIN T2 JOIN T3
    ON T1.a1 = T2.a2 AND T3.a3 = T2.b2
```

Jointure sur trois tables (il y a d'autres syntaxes possibles)

 La jointure se fait sur des attributs qu'il est préférable de préfixer par le nom de la table. Sinon, il pourrait y avoir ambigüité (si des attributs de tables différentes portent le même nom).

► **Opérations ensemblistes** :

```
SELECT ...
FROM ...
UNION
SELECT ...
FROM ...
```

Union

```
SELECT ...
FROM ...
INTERSECT
SELECT ...
FROM ...
```

Intersection

```
SELECT ...
FROM ...
EXCEPT
SELECT ...
FROM ...
```

Différence

– Il n'y a pas de parenthèses.
– La division cartésienne n'existe pas en SQL. Il faut se débrouiller autrement en reformulant la requête.
– On peut bien entendu faire un produit cartésien mais celui-ci n'a en général aucun sens pour une base de données.

► **Groupement** :

```
GROUP BY stock
HAVING stock > 10
```

```
GROUP BY nom, prenom
HAVING ...
```

```
GROUP BY auteur
HAVING COUNT(*) > 10
```

 Quand on opère un regroupement, les attributs dans le `SELECT` doivent être communs à tout le groupe!

► **Classement** :

```
ORDER BY prix (ASC par défaut)

ORDER BY stock DESC

ORDER BY note DESC, nom ASC (on trie déjà selon la note puis selon le nom)
```

► **Requêtes imbriquées** :

```
SELECT ...
FROM ...
WHERE a1 > (
            SELECT ...
           )
```

```
SELECT nom
FROM eleves
WHERE note = (
              SELECT MAX(note)
              FROM eleves
              )
```

 Ne pas oublier les parenthèses. L'indentation permet d'y voir clair.

► **Calcul** :

```
SELECT 1. / 3
```

```
SELECT
(SELECT SUM(note*coeff) FROM table)
/
(SELECT SUM(coeff) FROM table)
```

 Attention, si ce sont des entiers, on obtient une division euclidienne. Dans, l'exemple ci-dessus, si les notes et les coefficients sont entiers, on n'obtiendra pas le bon résultat. Le plus simple est alors d'écrire

```
SUM(note*coeff*1.)
```

pour forcer le type float.