TP SQL SYNTAXE

I. SELECT.

Ouvrir la base de données tomates et sélectionner la table tomate.

Pour obtenir le contenu d'une seule colonne de la table tomate :

```
SELECT masse
FROM tomate;
```

Pour obtenir le contenu de plusieurs colonnes de la table tomate :

```
SELECT identifiant, couleur
FROM tomate;
```

Pour obtenir toutes les colonnes de la table tomate :

```
SELECT *
FROM tomate;
```

II. DISTINCT.

Pour obtenir la liste des couleurs distinctes des tomates, sachant que deux tomates sont rouges :

```
SELECT DISTINCT(couleur)
FROM tomate;
```

Pour optimiser les performances il est préférable d'utiliser la commande SQL GROUP BY lorsque c'est possible. (Voir **IX**).

III. WHERE.

Pour avoir les tomates dont la masse est supérieur ou égale à 110 g :

```
SELECT *
FROM tomate
WHERE masse >= 110;
```

IV. AND et OR.

Pour obtenir les tomates dont le diamètre est supérieur à 50 mm et dont la couleur n'est pas rouge :

```
SELECT *
FROM tomate
WHERE diametre > 50 AND couleur != 'rouge';
```

Pour obtenir les tomates dont le diamètre est supérieur à 60 mm ou dont la couleur est rouge :

```
SELECT *
FROM tomate
WHERE diametre > 60 OR couleur = 'rouge';
```

Pour obtenir les tomates dont le diamètre est supérieur à 50 mm et dont la couleur est jaune ou verte :

```
SELECT *
```

```
FROM tomate
WHERE diametre > 50 AND (couleur = 'verte' OR couleur = 'rouge');
```

V. IN.

Pour obtenir les tomates dont la couleur est jaune ou verte.

```
SELECT *
FROM tomate
WHERE couleur IN ('jaune', 'verte' );
```

VI. BETWEEN.

Pour obtenir les tomates dont la masse est comprise entre 100 et 200 g.

```
SELECT *
FROM tomate
WHERE masse BETWEEN 100 AND 200;
```

VII. LIKE.

L'opérateur LIKE est utilisé dans la clause WHERE des requêtes SQL. Ce mot-clé permet d'effectuer une recherche sur un modèle particulier. Il est par exemple possible de rechercher les enregistrements dont la valeur d'une colonne commence par telle ou telle lettre. Les modèles de recherches sont multiples.

Pour obtenir toutes les tomates dont la couleur contient la lettre 'r'.

```
SELECT *
FROM tomate
WHERE couleur LIKE '%r%';
```

Pour obtenir toutes les tomates dont la masse est comprise entre 100 et 200 g exclu :

```
SELECT *
FROM tomate
WHERE masse LIKE '1__';
```

VIII. IS NULL / IS NOT NULL.

Pour obtenir les tomates dont la masse n'est pas renseignée :

```
SELECT *
FROM tomate
WHERE masse IS NULL;
```

Pour obtenir les tomates dont le diamètre est renseigné :

```
SELECT *
FROM tomate
WHERE masse IS NOT NULL;
```

IX. GROUP BY.

Pour obtenir la liste des couleurs distinctes des tomates, sachant que deux tomates sont rouges :

```
SELECT couleur
FROM tomate
GROUP BY couleur;
```

Cette commande doit toujours s'utiliser après la commande WHERE et avant la commande HAVING.

Pour obtenir la masse moyenne des tomates selon leur couleur :

```
SELECT couleur, AVG(masse)
FROM tomate
GROUP BY couleur;
```

Pour compter le nombre de tomates par couleur :

```
SELECT couleur, COUNT(*)
FROM tomate
GROUP BY couleur;
```

X. HAVING.

La condition HAVING est presque similaire à WHERE à la seule différence que HAVING permet de filtrer en utilisant des fonctions telles que **SUM()**, **COUNT()**, **AVG()**, **MIN()** ou **MAX()**.

```
SELECT attribut1, SUM(attribut2)
FROM nom_table
GROUP BY attribut1
HAVING fonction(attribut2) operateur valeur;
```

Cela permet donc de sélectionner les colonnes de la table "nom_table" en GROUPANT les lignes qui ont des valeurs identiques sur la colonne "attribut1" et que la condition de HAVING soit respectée.

Important : HAVING est très souvent utilisé en même temps que GROUP BY mais ce n'est pas obligatoire.

Pour obtenir la liste des masses moyennes des tomates selon leur couleur et dont la masse moyenne par couleur est supérieure à 100 g :

```
SELECT couleur, AVG(masse)
FROM tomate
GROUP BY couleur
HAVING AVG(masse) > 100;
```

XI. ORDER BY.

La commande ORDER BY permet de trier les lignes dans un résultat d'une requête SQL. Il est possible de trier les données sur une ou plusieurs colonnes, par ordre ascendant ou descendant.

```
SELECT attribut1, attribut2
FROM nom_table
```

```
ORDER BY attribut1;
```

Par défaut les résultats sont classés par ordre ascendant, toutefois il est possible d'inverser l'ordre en utilisant le suffixe DESC après le nom de la colonne. Par ailleurs, il est possible de trier sur plusieurs colonnes en les séparant par une virgule. Une requête plus élaborée ressemblerait à cela :

```
SELECT attribut1, attribut2, attribut3
FROM nom_table
ORDER BY attribut1 DESC,  attribut2 ASC;
```

XII. AS.

Il est possible d'utiliser des alias pour renommer temporairement une colonne ou une table dans une requête. Cette astuce est particulièrement utile pour faciliter la lecture des requêtes.

12.1 Alias sur une colonne.

Un alias permet de renommer le nom d'une colonne dans les résultats d'une requête SQL. C'est pratique pour avoir un nom facilement identifiable dans une application qui doit ensuite exploiter les résultats d'une recherche.

```
SELECT attribut1 AS nouveau_attribut1
FROM nom_table;
```

12.2 Alias sur une table.

Permet d'attribuer un autre nom à une table dans une requête SQL. Cela peut aider à avoir des noms plus court, plus simple et plus facilement compréhensible. Ceci est particulièrement vrai lorsqu'il y a des jointures.

```
SELECT attribut1, attribut2
FROM nom_table AS nouveau_nom_table;
```

XIII. Fonctions d'agrégation.

Les fonctions d'agrégation permettent d'effectuer des opérations statistiques sur un ensemble d'enregistrement. Étant données que ces fonctions s'appliquent à plusieurs lignes en même temps, elle permettent des opérations qui servent à récupérer l'enregistrement le plus petit, le plus grand ou bien encore de déterminer la valeur moyenne sur plusieurs enregistrements.

L'utilisation la plus générale consiste à utiliser la syntaxe suivante :

```
SELECT fonction(attribut) FROM nom_table;
```

13.1 AVG().

La fonction d'agrégation **AVG()** permet de calculer une valeur moyenne sur un ensemble d'enregistrement de type numérique et non nul.

```
SELECT AVG(attribut) FROM nom_table;
```

Cette requête permet de calculer la valeur moyenne de la colonne "attribut" sur tous les enregistrements de la table "nom_table". Il est possible de filtrer les enregistrements concernés à l'aide de la commande WHERE. Il est aussi possible d'utiliser la commande GROUP BY pour regrouper les données appartenant à la même entité.

13.2 COUNT().

Si on souhaite connaître le **nombre d'enregistrement** (les valeurs NULL ne sont pas comptabilisées) d'une colonne :

```
SELECT COUNT(attribut) FROM nom_table;
```

Si on souhaite connaître le **nombre total de lignes** (les valeurs NULL sont comptabilisées) de la table :

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

Enfin, il est également possible de compter le nombre d'enregistrement distinct pour une colonne. La fonction ne comptabilisera ni les doublons, ni les valeurs NULL pour une colonne choisie :

```
SELECT COUNT(DISTINCT attribut) FROM nom_table;
```

en général, en terme de performance il est plutôt conseillé de filtrer les lignes avec GROUP BY si c'est possible, puis d'effectuer un **COUNT**(*).

13.3 MAX().

La fonction d'agrégation **MAX()** permet de retourner la valeur maximale d'une colonne dans un set d'enregistrement. La fonction peut s'appliquée à des données numériques ou alphanumériques :

```
SELECT MAX(attribut) FROM nom_table;
```

Lorsque cette fonctionnalité est utilisée en association avec la commande GROUP BY, la requête peut ressembler à l'exemple ci-dessous:

```
SELECT attribut1, MAX(attribut2)
FROM nom_table
GROUP BY attribut1;
```

13.4 MIN().

La fonction d'agrégation **MIN()** permet de retourner la valeur minimale d'une colonne dans un set d'enregistrement. La fonction peut s'appliquée à des données numériques ou alphanumériques :

```
SELECT MIN(attribut) FROM nom_table;
```

Lorsque cette fonctionnalité est utilisée en association avec la commande GROUP BY, la requête peut ressembler à l'exemple ci-dessous:

```
SELECT attribut1, MIN(attribut2)
FROM nom_table
GROUP BY attribut1;
```

13.5 SUM().

La fonction d'agrégation **SUM()** permet de calculer la somme totale d'une colonne contenant des valeurs numériques. Cette fonction ne fonctionne que sur des colonnes de types numériques (INT, FLOAT …) et n'additionne pas les valeurs NULL :

```
SELECT SUM(attribut) FROM nom_table;
```

Il est possible de filtrer les enregistrements avec la commande WHERE pour ne calculer la somme que des éléments souhaités :

```
SELECT SUM(attribut2)
FROM nom_table
WHERE condition;
```

Il est possible de calculer la somme de chaque paramètre de la colonne 1 en groupant les lignes avec la commande GROUP BY :

```
SELECT attribut1, SUM(attribut2)
FROM nom_table
GROUP BY attribut1;
```