Travaux dirigés

Bases de données - SQL

Rendez vous sur le site http ://inloop.github.io/sqlite-viewer/ et téléchargez l'exemple demandé (pensez à le supprimer en fin de séance).

Requêtes SQL - premiers pas

1. Que renvoie l'instruction `SELECT * FROM 'Album'` ? et `SELECT title FROM 'Album' LIMIT 0,30` ?
2. **Renommage :** Que se passe-t-il lorsqu'on rajoute le mot-clé `AS` comme suit : `SELECT 'title' as 'Titre' FROM 'Album' LIMIT 0,30` ?
3. **Projection :** A l'aide de `SELECT DISTINCT`, obtenir les identifiants des artistes des 30 albums de plus petit identifiant sans redondance.
4. Que renvoie l'instruction `SELECT * FROM 'Album' LIMIT 0,30` ? Que rajoute `OFFSET 5` placé après la commande précédente ?
5. En utilisant des opérateurs parmi $+ - */, =, <>, <, <=, >, >=$,`AND`, `OR`, `NOT`, `IS NULL` et `IS NOT NULL`
   (a) **Sélection :** En choisissant la table `Track` et en utilisant le mot-clef `WHERE` ainsi qu'une expression logique adéquate, qui permet d'affiner la sélection, trouver les chansons dont l'identifiant d'album est plus grand que 12.
   (b) Obtenir les titres des chansons de la table `Track` dont l'initiale est B, C, D ou E.
   (c) Obtenir les titres des chansons dont l'initiale est 'B', et l'auteur a pour identifiant `12`
6. La commande `ORDER BY colonne` permet d'ordonner les résultats suivant les valeurs présentes dans colonne. Ajouter `ASC` ou `DESC` après le nom de la colonne spécifie si on souhaite faire un tri croissant ou décroissant. Comment récupérer les dix premiers titres de chansons commençant par 'B' dans l'ordre descendant de l'identifiant du groupe ?
7. **Fonctions d'agrégation :** Utiliser `MIN, MAX, SUM, AVG, COUNT` pour trouver le nombre de chansons dans la playliste de trackId 1 (table `PlaylistTrack`) ; les chansons ou albums dont le prix est inférieur à la moyenne des prix (table `Track`), les titres des chansons du groupe de plus grand identifiant. (*Exemple d'utilisation :* `SELECT COUNT(*) FROM` Album') Ces fonctions permettent d'utiliser des **agrégats**
8. **Sous-requêtes** On peut utiliser des requêtes imbriquées ; on parle alors de sous-requête pour celle qui est utilisée par l'autre. On délimite une sous-requête par des parenthèses, et on la place après `FROM` à la place d'une table, ou au sein d'une expression logique (en suite d'un `Where`).
   Chercher le titre de l'avant-dernier album de la table album. Obtenir le nom de l'artiste concerné (en une requête).
9. **Jointures** La jointure permet de sélectionner les enregistrements comportant ceux de deux tables satisfaisant une même contrainte logique. Ainsi on trouvera dans la relation `SELECT * FROM utilisateur JOIN adresse ON tableUtilisateurs.id= tableAdresse.id` les enregistrement comportant les informations sur les utiisateurs, y compris celles relatives à leur adresse.
   - Dans l'exemple cité, la condition est une condition d'égalité.
   - La commande `INNER JOIN` renvoie les enregistrements pour lesquels on obtient une égalité liant deux tables
     Obtenir les nom des albums avec le nom des groupes ayant produit ces albums, puis le nom de leur genre. La commande `JOIN` est une version abrégée de `INNER JOIN`.

- La jointure externe `LEFT JOIN` renvoie l'ensemble des enregistrements de la table de gauche, même si la condition ne permet pas d'obtenir un résultat dans l'autre table.

10. La commande `GROUP BY` permet de regrouper les résultats d'une requête. Un ensemble de livres peut par exemple être regroupé par auteur, pour obtenir le nombre total de pages écrites par chaque auteur. On utilisera la syntaxe

    ```
    SELECT auteur, SUM(livres) FROM bibliotheque GROUP BY auteur
    ```

11. La commande `HAVING`, proche de `WHERE`, permet de filtrer en utilisant des fonctions (`SUM()`, `COUNT()`, etc.). Elle suit la syntaxe

    ```
    SELECT a, SUM(b)
    FROM table
    GROUP BY a
    HAVING f(b)>12
    ```

    On groupe les lignes qui ont les mêmes valeurs en colonne a, la condition portant ici sur la colonne b.

    Trouver les identifiants des auteurs ayant plusieurs chansons à leur actif.

12. **Opérateurs ensemblistes :**
    - **UNION** : `UNION SELECT (*) FROM table1, table2, table2`
    - **INTERSECT** : `SELECT a FROM table 1 INTERSECT SELECT b FROM table2`. Peut être remplacé par `SELECT a FROM table1 WHERE a IN (SELECT b FROM table2)`
    - **EXCEPT** : `SELECT a FROM table 1 EXCEPT SELECT b FROM table2`. Peut être remplacé par `SELECT a FROM table1 WHERE a NOT IN (SELECT b FROM table2)`
    - **produit cartésien** : `SELECT a,b, FROM table1, table2`

☞Les commandes doivent s'utiliser dans un ordre suivant (lorsqu'on en utilise plusieurs d'entre elles) :

```
SELECT *
FROM table
WHERE condition
GROUP BY expression
HAVING condition
{ UNION | INTERSECT | EXCEPT }
ORDER BY expression
LIMIT count
OFFSET start
```

Seuls `SELECT` et `FROM` sont indispensables.