BDD SQL – Séance 2

Rappel / Pour commencer :

- Lancez **SQLiteStudio** et ouvrez la base de données **chinook.db**, si elle n'est pas encore ouverte.
  - **Affichage → Cocher « Base de données »** ou **View → Cocher « Database »** en anglais pour afficher le volet gauche
  - Si ça dit **chinook**, la BDD est déjà ouverte
  - Sinon, ouvrez-la à travers **Base-de-données → Ajouter une base-de-données** ou **Database → Add Database** en anglais.
- Pour interroger la BDD, on ouvre **l'éditeur SQL** (Alt+E ou **Outils → Ouvrir l'éditeur SQL**), entre des requêtes au champs **Requête**, et exécute-les en cliquant sur le triangle bleu, ou avec le raccourci **F9**.
- Pour plus de détails, regardez l'énoncé du dernier devoir.
- La dernière fois, on a vu des requêtes basiques de filtrage et de trie. Par exemple :
  - La requête
    ```
    SELECT * FROM customers WHERE lastname = "Tremblay";
    ```
    affiche tous les entrées de la table **customers** dont le valeur du champ **LastName** est "Tremblay"
  - La requête
    ```
    SELECT * FROM customers WHERE lastname LIKE "Tremblay";
    ```
    est pareil, mais *insensitive à la case*, ce qui signifie qu'elle ne fait pas de distinction entre les majuscules et les minuscules. Dans ce cas, la requete rendrait aussi les entrées dont le nom est par exemple "TREMBLAY" ou "tremblay"
  - L'utilisation de la caractère **"%"** dans les requêtes permet de chercher des entrées dont la valeur d'un attribut correspond à un *modèle* (anglais : *pattern*), où le "%" représente n'importe quelle séquence de caractères (possiblement vide).
    Par exemple, la requête
    ```
    SELECT * FROM customers WHERE lastname = "a%n%";
    ```
    rendre toutes les entrées de la table où l'attribut **LastName** commence avec un "a" ou "A", et contient un "n" ou "N".

Comment rendre vos devoirs

- Ecrivez vos réponses dans **Bloc-notes** (ou **Notepad++**, si **Bloc-notes** ne marche pas)**,** et sauvegardez-le comme **NOM-SQL-TD2.txt,** où vous mettez votre nom pour **NOM**.
- Pour chaque question, mettez une ligne contenant uniquement **deux tirets,** suivi du numéro de l'exercice, et mettez votre réponse après. Par exemple, les premières lignes du fichier – correspondant aux questions 1 et 2 en bas – seront

```
-- 1
SELECT …

-- 2
SELECT …
…
```

Attention : Si la réponse est de la forme d'une requête "`SELECT …`", tapez-la d'abord dans l'éditeur SQL, pour vérifier si ça marche. Puis, copiez-collez la requête dans Bloc-notes.

Quand vous aurez finis, déposez le fichier sur Teams, ou envoyez-le par email si vous ne pouvez pas acceder à teams.

Exercices de révision

Voici des exercices dans le style de la séance de la semaine dernière pour commencer ; veuillez consulter la première feuille d'exercices pour les explications sur les constructions de requêtes SQL pertinentes.

Exercice 1. Donnez une requête pour afficher le `LastName` et l'`Email` des clients de la table `customers` dont le prénom (`FirstName`) commence par la lettre « **J** » et l'e-mail contient « **@yahoo.com** ».

Exercice 2. Donnez une requête pour afficher les différentes valeurs de la colonne `BillingCountry` de la table `invoices`, triées par ordre alphabétique inverse (décroissant).

Exercice 3. Donnez une requête pour afficher l'`InvoiceId` et le `Total` des factures dont le total est **strictement entre $5.00 et $10.00** (utilisez `AND`).,

Exercice 4. Donnez une requête qui affiche le `Name` de la table `tracks` ainsi que sa taille en **Mégabytes (MB)** (en utilisant la colonne **Bytes**). Arrondissez le résultat à **quatre décimales** et renommez la colonne `SizeMB`. (Rappel : 1 Mo = 1,048,576 octets).
Astuce : On utilise the mot clé `AS` pour renommer des colonnes. Voir l'explication avant de l'exercice 34 sur la première feuille d'exercices.

Exercice 5. Donnez une requête pour afficher uniquement les entrées de la table `albums` dont le titre (`Title`) contient le mot « **Greatest** » mais ne commence pas par le mot « **The** ».

Jointures

Parfois, il est nécessaire d'effectuer des requêtes sur plusieurs tables simultanément. Par exemple, imaginons que l'on veuille connaître les noms des clients ayant passé des commandes d'une valeur totale de 0,99. Le problème est que la table `invoices` ne contient pas les noms des clients, mais seulement une colonne `customerid`. La solution consiste à combiner les tables `invoices` et `customers` en les **joignant** via l'attribut `customerid`, à l'aide de la requête suivante :

```
SELECT customers.FirstName, customers.LastName
  FROM customers INNER JOIN invoices
    ON customers.customerid = invoices.customerid
 WHERE invoices.TOTAL = 0.99;
```

Typiquement, les jointures sont effectuées en utilisant des paires d'une **clé primaire** dans une table, et une **clé étrangère** correspondante dans une autre table. Par exemple, dans l'exemple ci-dessus, `customerid` est la **clé primaire** de la table `customers` — ce qui signifie qu'elle identifie de manière **unique** ses entrées — et elle est une **clé étrangère** dans la table `invoices`, établissant ainsi le lien entre la table `invoices` et la table `customers`.
L'information concernant si une colonne contient une clé primaire ou étrangère est accessible en double-cliquant sur les noms des colonnes dans la barre latérale (activée via **Affichage → Base de données**).

Exercices sur jointures

Les questions suivantes doivent être répondues en utilisant des **jointures**, avec uniquement les techniques qu'on a apprises aujourd'hui et dans la dernière séance. Astuce : vous pouvez utiliser le **schéma de la base de données** sur la dernière page pour trouver les clés sur lesquelles il faut joindre les tables.

Exercice 6. Écrivez une requête SQL qui affiche le nom de chaque piste (`tracks.Name`) et le titre de l'album (`albums.Title`) auquel elle appartient.

Exercice 7. Écrivez une requête SQL qui affiche le titre de chaque album (`albums.Title`) ainsi que le nom de l'artiste (`artists.Name`) qui a créé l'album.

Exercice 8. Écrivez une requête SQL qui affiche le nom complet de chaque client (`customers.FirstName`, `customers.LastName`) ainsi que le nom complet de leur représentant commercial (`employees.FirstName`, `employees.LastName`).

Exercice 9. Écrivez une requête SQL qui affiche le nom de la piste (`tracks.Name`) et le nom du type de média (`media_types.Name`) de cette piste.

Exercice 10. Écrivez une requête SQL qui affiche l'identifiant de la facture (`invoices.InvoiceId`) et le nom complet du client qui a effectué l'achat (`customers.FirstName`, `customers.LastName`). Triez par `InvoiceId` croissant.

Exercice 11. Écrivez une requête SQL pour afficher les noms des albums de l'artiste **Led Zeppelin**. (Utiliser les tables `albums` et `artists` et la clé `ArtistId`)

Exercice 12. Écrivez une requête SQL qui affiche le titre des chansons (`tracks.Name`) et le titre de leurs albums (`albums.Title`).
Ensuite, renommez la colonne `Name` en `TrackName` et la colonne `Title` en `Albumtitle`.

Exercice 13. Écrivez une requête SQL qui affiche les noms des clients (`customers.FirstName`, `customers.LastName`) ainsi que les totaux de leurs factures (`invoices.Total`). Triez les résultats par `total` décroissant.

Exercice 14. Écrivez une requête SQL pour afficher les titres de chansons (`tracks.Name`) et les noms de leurs genres (`genres.Name`).

Exercice 15. (**Difficile**) Écrivez une requête SQL qui affiche le prénom et le nom des employés (`employees.FirstName`, `employees.LastName`) ayant effectué des ventes, ainsi que les totaux des ventes correspondants (`invoices.Total`).
Indice : Cet exercice nécessite un « double join ». Jetez un œil au **schéma de la base de données** à la dernière page.

Fonctions d'agregation

Les **fonctions d'agrégation** permettent de résumer ou de calculer des informations sur plusieurs lignes de données. Voici les plus courantes :

- `SELECT COUNT(*) FROM Customers;` compte le nombre de clients dans la table `customers`.
- `SELECT SUM(Total) FROM Invoices;` calcule la somme des valeurs d'une colonne numérique.
- `SELECT AVG(UnitPrice) FROM Tracks;` calcule la moyenne des valeurs d'une colonne numérique.
- `SELECT MIN(UnitPrice) FROM Tracks;` trouve la valeur minimale d'une colonne.
- `SELECT MAX(UnitPrice) FROM Tracks;` trouve la valeur maximale d'une colonne.

Exercices – fonctions d'agregation

Exercice 16. Écrivez une requête SQL qui renvoie la vente la plus élevée (c'est-à-dire la valeur la plus élevée dans la colonne `Total` de la table `Invoices`).

Exercice 17. Écrivez une requête SQL qui renvoie le nombre total de `tracks` par le groupe **AC/DC**.

Exercice 18. Écrivez une requête SQL qui renvoie la longueur moyenne en millisecondes des `tracks` de genre `Rock`.

Fonctions d'agrégation avec "group by"

Dans SQL, l'agrégation avec `GROUP BY` permet de regrouper les lignes d'une table en fonction de certaines colonnes, puis d'appliquer des fonctions d'agrégation (comme COUNT(), SUM(), AVG(), etc.) sur ces groupes. Cela est utile lorsqu'on souhaite effectuer des calculs ou obtenir des statistiques sur des sous-ensembles de données.

Exemple : la requête

- ```
  SELECT CustomerId, SUM(Total) AS MontantTotalVentes
  FROM Invoices
  GROUP BY CustomerId;
  ```

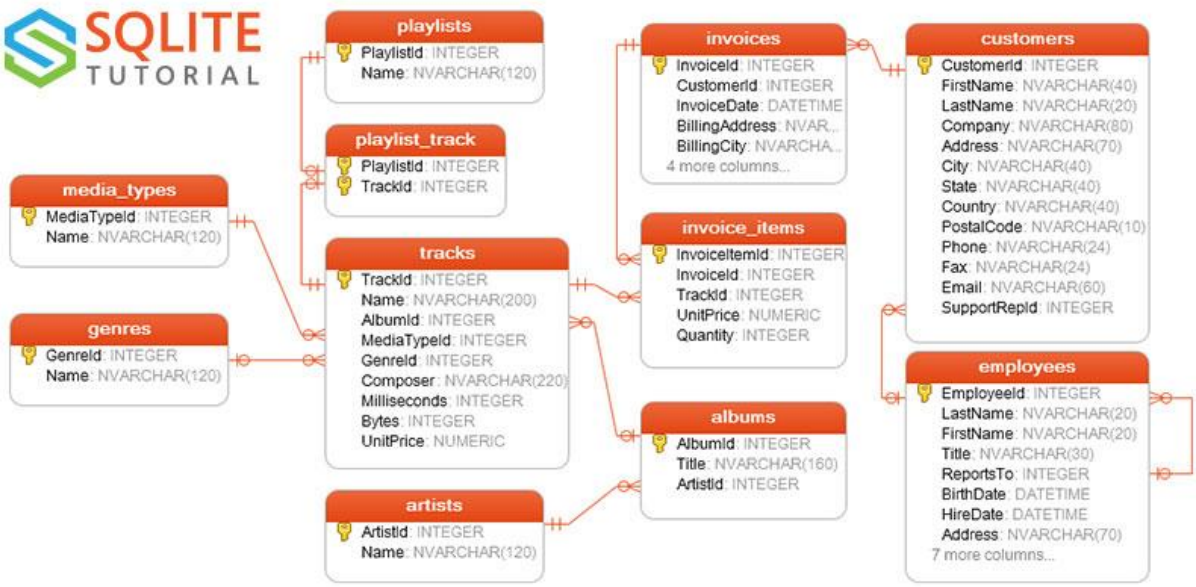
renvoie le montant total des ventes pour chaque client.

Exercices

Exercice 19. Écrivez une requête SQL qui renvoie le nombre de morceaux (`TrackId`) pour chaque genre de musique dans la table `Genres`.

Exercice 20. Écrivez une requête SQL qui renvoie le nombre de clients pour chaque pays dans la table `Customers`.

Exercice 21. Écrivez une requête SQL qui renvoie le nombre de pistes pour chaque album, en joignant les tables `Albums` et `Tracks`.
  ```


SQLITE TUTORIAL
playlists
PlaylistId: INTEGER
Name: NVARCHAR(120)
playlist_track
PlaylistId: INTEGER
TrackId: INTEGER
media_types
MediaTypeId: INTEGER
Name: NVARCHAR(120)
genres
GenreId: INTEGER
Name: NVARCHAR(120)
tracks
TrackId: INTEGER
Name: NVARCHAR(200)
AlbumId: INTEGER
MediaTypeId: INTEGER
GenreId: INTEGER
Composer: NVARCHAR(220)
Milliseconds: INTEGER
Bytes: INTEGER
UnitPrice: NUMERIC
artists
ArtistId: INTEGER
Name: NVARCHAR(120)
invoices
InvoiceId: INTEGER
CustomerId: INTEGER
InvoiceDate: DATETIME
BillingAddress: NVAR...
BillingCity: NVARCHA...
4 more columns...
invoice_items
InvoiceItemId: INTEGER
InvoiceId: INTEGER
TrackId: INTEGER
UnitPrice: NUMERIC
Quantity: INTEGER
albums
AlbumId: INTEGER
Title: NVARCHAR(160)
ArtistId: INTEGER
customers
CustomerId: INTEGER
FirstName: NVARCHAR(40)
LastName: NVARCHAR(20)
Company: NVARCHAR(80)
Address: NVARCHAR(70)
City: NVARCHAR(40)
State: NVARCHAR(40)
Country: NVARCHAR(40)
PostalCode: NVARCHAR(10)
Phone: NVARCHAR(24)
Fax: NVARCHAR(24)
Email: NVARCHAR(60)
SupportRepId: INTEGER
employees
EmployeeId: INTEGER
LastName: NVARCHAR(20)
FirstName: NVARCHAR(20)
Title: NVARCHAR(30)
ReportsTo: INTEGER
BirthDate: DATETIME
HireDate: DATETIME
Address: NVARCHAR(70)
7 more columns...