2 - SQL pas à pas

1 Introduction

LOGILIVRES, une entreprise de logistique dédiée à la distribution de livres pour des librairies, souhaite mettre à jour ses fichiers clients (qui sont sur tableur à l'heure actuelle) en utilisant des bases de données relationnelles afin d'améliorer l'organisation de l'information.
La base de données devra gérer :

- la liste des librairies clientes ;
- la liste des livres gérés par l'entreprise ;
- la liste des commandes des librairies.

L'équipe chargée de la création de la base de données a ainsi défini le dictionnaire des données suivant :

Nom	Signification	Type
NumLib	Identifiant de la librairie	Numérique
NomLib	Nom de la librairie	Texte
AdresseLib	Adresse de la librairie	Texte
TelLib	Téléphone de la librairie	Alphanumérique
NumLivre	Identifiant du livre	Numérique
AutLivre	Auteurs du livre	Texte
EditeurLivre	Editeur du livre	Texte
PrixLivre	Prix actuel du livre en euros	Numérique
NumCommande	Identifiant de la commande	Numérique
DateCommande	Date où la commande est passée	Date
QuantiteLivre	Nombre d'éléments commandés	Numérique

Tracer le schéma relationnel de cette base de données :

2 Création et modification de bases de données

Le langage SQL (*Structured Query Language*) permet la **définition de tables** (création, modification ou suppression), la **manipulation de données** (extractions, ajouts, modifications ou suppressions de contenus) et leur **protection** (définition des permissions).

Grâce à ce langage, le SI fournit aux différents acteurs d'une organisation des données précises, exhaustives et à jour.
Les instructions de traitement formulées en SQL sont des **requêtes**, autrement dit des demandes de traitement.

Sources sur le SQL :

- https://sql.sh/cours
- https://www.w3schools.com/sql/default.asp
- https://sql.developpez.com
- Pour coder en SQL : https://basthon.fr

2.1 Création de tables

La commande CREATE TABLE permet de créer une table en SQL.
Une table est une entité qui est contenue dans une base de données pour stocker des données ordonnées dans des colonnes.
La création d'une table sert à définir les colonnes et le type de données qui seront contenus dans chacune des colonne (entier INT, chaîne de caractères TEXT, date DATE, valeur binaire BOOLEAN, nombre décimal NUMERIC ...).

```
CREATE TABLE nom_de_la_table
(
    colonne1 type_donnees,
    colonne2 type_donnees,
    colonne3 type_donnees,
    colonne4 type_donnees
)
```

Ecrire la requête permettant de créer la Table *Librairies* :

.....

.....

.....

.....

.....

2.2 Ajout d'entrées

L'insertion de données dans une table s'effectue à l'aide de la commande INSERT INTO. Cette commande permet au choix d'inclure une seule ligne à la base existante ou plusieurs lignes d'un coup :

INSERT INTO table VALUES ('valeur 1', 'valeur 2', ...)

On peut également choisir d'ajouter des entrées avec seulement certaines colonnes remplies avec la syntaxe :

INSERT INTO table (nom_colonne_1, nom_colonne_2, ...)
VALUES ('valeur 1', 'valeur 2', ...)

Ecrire la requête permettant d'ajouter dans la table *Librairies* les entrées suivantes :

- Chroniques, 19 rue Camille Desmoulins, 94230 Cachan, 01 41 98 62 62
- Fontaine, 88 Rue de Sèvres, 75007 Paris, 01 47 83 29 71
- Shakespeare and Company, 37 Rue de la Bûcherie, 75005 Paris, 01 43 25 40 93

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

2.3 Afficher toutes les entrées d'une table

L'utilisation la plus courante de SQL consiste à lire des données issues de la base de données. Cela s'effectue grâce à la commande SELECT, qui retourne des enregistrements dans un tableau de résultat. Cette commande peut sélectionner une ou plusieurs colonnes d'une table.
On peut utiliser la requête suivante pour afficher toutes les colonnes d'une table :

SELECT * FROM table

Créer les autres tables et les remplir avec des données.

2.4 Modifier et supprimer une entrée

La commande UPDATE permet d'effectuer des modifications sur des lignes existantes.
Très souvent cette commande est utilisée avec WHERE pour spécifier sur quelles lignes doivent porter la ou les modifications.

```
UPDATE table
SET colonne_1 = 'valeur 1', colonne_2 = 'valeur 2', colonne_3 = 'valeur 3'
WHERE condition
```

La Librairie Chroniques s'est rendu compte que son nom a été mal renseigné dans la base de données et demande à ce qu'il soit changer en **"Librairie Chroniques"**.
Ecrire la requête permettant de modifier cette entrée :

. . . . . .

. . . . . .

. . . . . .

La commande DELETE en SQL permet de supprimer des lignes dans une table.
En utilisant cette commande associé à WHERE il est possible de sélectionner les lignes concernées qui seront supprimées.

```
DELETE FROM table
WHERE condition
```

La librairie Shakespeare and Company souhaite changer de prestataire logistique et arrête son contrat avec LOGILIVRES.
Ecrire la requête permettant de supprimer cette entrée :

. . . . . .

. . . . . .

3 Extraction des données d'une table

Pour extraire des données d'une table, on peut utiliser la commande SELECT qui a été utilisé dans le 2.3. Les différents mots clés sont alors :

SELECT	-> pour indiquer le nom des colonnes à afficher
FROM	-> pour indiquer les tables mobilisées
WHERE	-> pour indiquer une condition sur les lignes à afficher
AND	-> pour ajouter d'autres conditions sur les lignes à afficher
ORDER BY	-> pour trier les lignes

Exemple : On peut utiliser cette requête pour afficher les livres dont le prix est supérieur à 10 € et les afficher par ordre alphabétique des noms d'éditeur :

```
SELECT EditeurLivre, AuteurLivre, NomLivre, PrixLivre
FROM Livres
WHERE PrixLivre > 10
ORDER BY EditeurLivre
```

EditeurLivre	AuteurLivre	NomLivre	PrixLivre
Dunod	Hudin Oona	Système d'information de gestion	39.9
Dunod	Acker, Agnès	Astronomie Astrophysique	45
Livre de Poche	Sanderson, Brandon	La Voie des Rois	11.9
Nathan	Darlay, Christine	Processus 7	24.9
Storyline	Cordova, Jason	Brindlewood Bay	35

Ecrire la requête permettant d'afficher le nom et l'adresse des librairies dans l'ordre alphabétique :

. . . . . .

. . . . . .

. . . . . .

. . . . . .

4 Extraction des données de plusieurs tables

Pour pouvoir extraire des données de plusieurs tables, il va falloir utiliser ce qu'on nomme des jointures.
Elles permettent de créer de nouvelles tables combinant deux tables différentes (on peut même faire des jointures avec plusieurs tables à la fois).

```
SELECT *
FROM table_1
JOIN table_2 ON table_1.key = table_2.key
```

Exemple : On peut utiliser cette requête pour afficher les commandes datant d'avant le mois d'Octobre, leurs dates et les noms et l'adresses des librairies concernées :

```
SELECT Commandes.NumCommande, Librairies.NomLib, Commandes.DateCommande, Librairies.AdresseLib
FROM Commandes
JOIN Librairies ON Commandes.NumCli = Librairies.NumLib
WHERE Commandes.DateCommande < "2025-10-01" ;
```

NumCommande	NomLib	DateCommande	AdresseLib
1	Les rois du bouquin	2025-09-23	42 avenue du Beffroi, 59300 Dunkerque
2	La Mare aux livres	2025-09-30	3 rue des Clouteries, 62500 Saint Omer

Ecrire la requête permettant d'afficher le numéro de la commande, les livres commandés, leurs prix et leur quantité :

. . . . . .

. . . . . .

. . . . . .

Exemple : On peut utiliser cette requête pour afficher les commandes datant d'avant le mois d'Octobre, leurs dates et les noms et l'adresses des librairies concernées ainsi que les identifiants et la quantité de livres commandés :

```
SELECT Commandes.NumCommande, Librairies.NomLib, Commandes.DateCommande, Librairies.AdresseLib,
       CommandesLivres.NumLivre, CommandesLivres.QuantiteLivre
FROM Commandes
JOIN Librairies ON Commandes.NumCli = Librairies.NumLib
JOIN CommandesLivres ON Commandes.NumCommande = CommandesLivres.NumCommande
WHERE Commandes.DateCommande < "2025-10-01" ;
```

NumCommande	NomLib	DateCommande	AdresseLib	NumLivre	QuantiteLivre
1	Les rois du bouquin	2025-09-23	42 avenue du Beffroi, 59300 Dunkerque	1	10
1	Les rois du bouquin	2025-09-23	42 avenue du Beffroi, 59300 Dunkerque	3	20
2	La Mare aux livres	2025-09-30	3 rue des Clouteries, 62500 Saint Omer	1	5
2	La Mare aux livres	2025-09-30	3 rue des Clouteries, 62500 Saint Omer	2	15

Ecrire la requête permettant d'afficher le numéro et la date de la commande, le nom et l'adresse de la librairie, les livres commandés ainsi que leurs prix et leur quantité :

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

5 Fonctions d'agrégation

Les fonctions d'agrégation dans le langage SQL permettent d'effectuer des opérations statistiques sur un ensemble d'entrées.
Les principales fonctions d'agrégation sont :

COUNT(*colonne*)	-> permet de compter le nombre d'entrées non *NULL* de la colonne
MIN(*colonne*)	-> renvoie le minimum de la colonne
MAX(*colonne*)	-> renvoie le maximum de la colonne
SUM(*colonne*)	-> renvoie la somme des entrées de la colonne
AVG(*colonne*)	-> renvoie la moyenne des entrées de la colonne
GROUP_CONCAT(*colonne*)	-> concatène les textes de la colonne

Exemples :

• On peut utiliser cette requête pour compter le nombre d'éléments de la table Livres :

```
SELECT COUNT(*) FROM Livres
```

COUNT(*)
6

• Cette requête permet de calculer le prix moyen des livres :

```
SELECT AVG(PrixLivre) AS "PrixMoyen"
FROM LIVRES
```

PrixMoyen
27.683333333333334

• On peut utiliser la fonction ROUND() pour arrondir le prix moyen :

```
SELECT ROUND(AVG(PrixLivre),2) AS "PrixMoyen"
FROM LIVRES
```

PrixMoyen
27.68

À l'aide de la fonction SUM(), créer une requête pour compter le nombre de livres des 2 premières commandes :

. . . . . .

. . . . . .

. . . . . .

Si on veut restreindre l'application des fonctions d'agrégation à certains paramètres, on peut utiliser la clause GROUP BY.
La clause SQL GROUP BY sépare l'ensemble de données en groupes sur la base de valeurs correspondantes dans une ou plusieurs colonnes, ce qui permet d'appliquer des fonctions d'agrégation à chaque groupe indépendamment.
GROUP BY est très utile quand on souhaite utiliser les fonctions d'agrégation à partir de jointures.

Exemple : la requête suivante permet de calculer le prix de vente pour chaque livres et chaque commande :

```
SELECT NumCommande, Livres.NomLivre, QuantiteLivre, SUM(Livres.PrixLivre*QuantiteLivre) AS "PrixTotal"
FROM CommandesLivres
JOIN Livres ON CommandesLivres.NumLivre = Livres.NumLivre
GROUP BY NumCommande, Livres.NumLivre
```

NumCommande	NomLivre	QuantiteLivre	PrixTotal
1	La Voie des Rois	10	119
1	Brindlewood Bay	20	700
2	La Voie des Rois	5	59.5
2	La Passe Miroir	15	141
3	Système d'information de gestion	10	399
3	Processus 7	15	373.5
4	Astronomie Astrophysique	10	450

Créer une requête afin d'afficher le numéro et la date de la commande, le nom et l'adresse de la boutique qui passe la commande et à l'aide de fonctions d'agrégation, la liste des livres commandés et le coût de la commande :

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Applications

1 - Prix Nobel :

1. Télécharger la base de données des Prix Nobel et l'ouvrir avec l'ouvrir avec Basthon ou un autre outil de votre choix.
   Pour prendre connaissance des attributs et de leur signification, saisir SELECT * FROM nobel LIMIT 10.
2. Écrire les requêtes SQL permettant :
   (a) d'obtenir les catégories dans lesquelles sont attribuées les prix Nobel ;
   (b) de lister par ordre alphabétique (sans doublon) les lauréats du prix Nobel nés en France ou travaillant pour une organisation Française ;
   (c) de lister les années où le "Comité international de la Croix Rouge" a obtenu le prix Nobel ;
   (d) de connaître le nombre de femmes ayant obtenu un prix Nobel ;
   (e) de lister les femmes françaises ayant obtenu un prix Nobel depuis 2005 ainsi que la catégorie de ce prix Nobel ;
   (f) de lister par âge décroissant les lauréats individuels du prix Nobel qui sont toujours en vie (leur DeathYear est alors NULL) ;
   (g) de rechercher les lauréats dont le nom contient "Curie".

2 - Basketball :

On va utiliser les fonctions d'agrégation de SQL afin d'obtenir de nouvelles informations sur les joueurs de la NBA.

1. Télécharger la base de données des joueurs de NBA (saison 2025-2026) et l'ouvrir avec Basthon ou un autre outil de votre choix.
   Cette base de données est composée d'une seule table, dont le schéma relationnel est le suivant :
   joueurs_nba (**Name**, Team, Number, Position, Age, Height, Weight)
   La taille (Height) d'un joueur est donné en cm et son poids (Weight) est donné en kg. L'attribut Number correspond au numéro que le joueur porte sur son maillot et l'attribut Position est l'abréviation de son poste sur un terrain de basket.
   Pour visualiser les premiers enregistrements de cette relation, saisir SELECT * FROM joueurs_nba.
2. Tester la requête SQL suivante et exprimer en français de tous les jours la nature de cette requête.
   SELECT AVG(Age) FROM joueurs_nba WHERE Team = "Los Angeles Lakers" ;
3. Écrire les requêtes SQL permettant :
   (a) d'obtenir la taille maximum des joueurs de l'équipe des Chicago Bulls ;
   (b) d'obtenir le poids le plus faible de tous les joueurs de NBA évoluant au poste de Shooting Guard ;
   (c) d'obtenir toutes les informations concernant Victor Wembanyama puis de savoir s'il est le plus jeune des joueurs de NBA de plus de 2m10 (on peut faire deux requêtes séparées) ;
4. Modification de la base de données
   (a) Écrire la requête SQL permettant de vous ajouter dans la base des données des joueurs de NBA, au sein de l'équipe des Boston Celtics avec le numéro 99 et au poste de Power Forward (ou bien tout autre valeur de votre choix) ;
   Vérifier ensuite que l'entrée a bien été ajoutée.

(b) On suppose que Nicolas Batum, qui évolue dans l'équipe des Los Angeles Clippers, va rejoindre Victor Wembanyama chez les San Antonio Spurs où il portera alors le numéro 75. Écrire la ou les requête(s) qui permet(tent) de mettre à jour la base de données.
Vérifier ensuite par une requête du type SELECT * FROM joueurs_nba WHERE Name = 'Nicolas Batum' que la mise à jour a bien été effectuée.

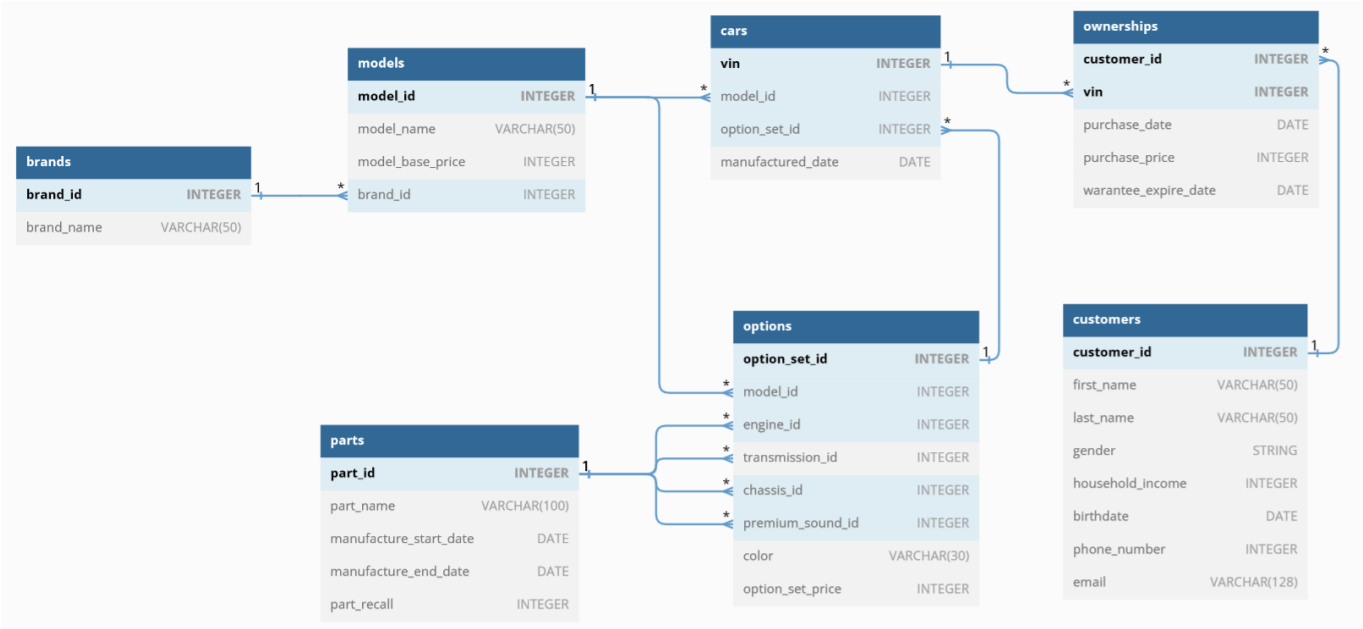
(c) Un plaisantin a pollué la table de données en insérant une équipe Gotham City. Écrire la requête SQL permettant de connaître les noms des joueurs de cette équipe, puis écrire la requête permettant de supprimer ces enregistrements.

3 - À chacun sa voiture :

On s'intéresse ici à la base de données car_database regroupant diverses informations sur des voitures :

- marque et modèle,
- numéro d'immatriculation (vehicle identification number, vin dans les tables),
- année de fabrication,
- nom du propriétaire,
- options,
- etc...

Les différentes tables sont représentées ci-dessous :



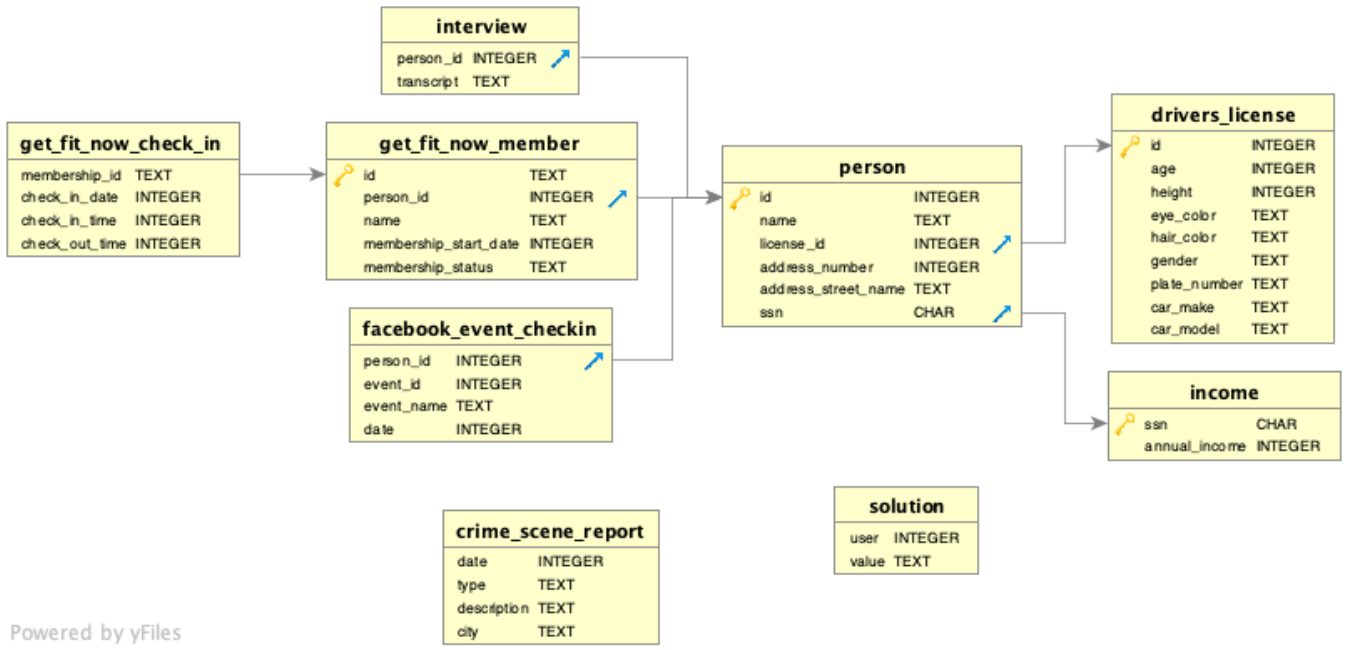
Sur cette figure :

- chaque tableau correspond à une table dont le nom est indiqué sur la première ligne ;
- les lignes suivantes listent les attributs et leur type. VARCHAR(255) signifie que l'attribut est un texte de 255 caractères au maximum ;
- les clés primaires de chaque table sont indiquées en gras. Notez que la table ownerships a une clé primaire multiple, à savoir le couple (customer_id, vin).
- les clés étrangères sont représentées par des liaisons entre les tables.

Cette base est téléchargeable ici. Les requêtes à effectuer sont données dans ce carnet Jupyter, à ouvrir avec basthon.fr.

4 - Enquête policière :

Le but de cette activité, proposé sur le site de l'université américaine Northwerstern University, est de résoudre un crime en utilisant des requêtes SQL dans une base de données dont voici le schéma relationnel :



> *Traduction approximative de la description de l'activité* : un crime a eu lieu et le détective a besoin de votre aide. Le détective vous a donné le rapport de la scène de crime mais vous l'avez perdu. Vous vous souvenez vaguement que le crime est un assassinat ayant eu lieu le 15 janvier 2018 à SQL City. La première étape est de retrouver le rapport de scène de crime à partir de la base de données de la police. Tous les indices sont enfouis dans cette base de données.

Toutes les requêtes peuvent être exécutées directement sur le site ici ; cependant vous pouvez télécharger la base de données afin d'exécuter vos requêtes en local.
À vous de trouver le coupable !