NSI - Cours et exercices sur le langage SQL

Auteurs : Christophe LIGERET (lycée Georges Clemenceau, Villemomble) et
David YANA (lycée Gustave Eiffel, Gagny)

Niveau : Terminale

Spécialité : Numérique et Sciences Informatiques (NSI)

Juin 2020

Extrait du BO :

Langage SQL : requêtes d'interrogation et de mise à jour d'une base de données.	Identifier les composants d'une requête. Construire des requêtes d'interrogation à l'aide des clauses du langage SQL : SELECT, FROM, WHERE, JOIN. Construire des requêtes d'insertion et de mise à jour à l'aide de : UPDATE, INSERT, DELETE.	On peut utiliser DISTINCT, ORDER BY ou les fonctions d'agrégation sans utiliser les clauses GROUP BY et HAVING.

Sommaire

I/ Introduction

Le langage SQL (*Structured Query Language :* en français **langage de requête structurée)** est un langage informatique destiné à travailler sur des bases de données.

Ce langage permet de :
1°) Créer / modifier des données dans une base de données
2°) Manipuler les données des bases de données : par exemple avec des fonctions de recherche ; d'ajout / modification / supression de données.

SQL a été créé en 1974 puis normalisé en 1986 ; ce langage est reconnu par la grande majorité des systèmes de gestion de bases de données relationnelles (abrégé SGBDR) existants.

Les scientifiques Edgar Frank Codd ; Donald Chamberlin et Raymond Boyce sont considérés comme les pères fondateurs des bases de données structurées et du langage SQL.

Dans ce document, nous allons utiliser le logiciel gratuit "DB Browser for SQLite". Il nous permettra de créer des bases de donnéess et effectuer des requêtes sur ces dernières : "DB Browser for SQLite" est téléchargeable à cette adresse : *https://sqlitebrowser.org/blog/first-release-candidate-for-3-12-0/*.

II/ Création et modifications de bases de données avec SQL

II.1/ Création d'un base de données

La création d'une base de données en SQL est possible en ligne de commande avec l'instruction du type :

`CREATE DATABASE ma_base_de_données` ou encore `CREATE TABLE ma_base_de_donnée`, selon le SGBD utilisé : étudier la documentation technique du SGBD !

Notons toutefois que la plupart des SGBD permettent de créer des bases de données avec des interfaces graphiques conviviales : ci-contre : exemple de définition d'une base de données avec

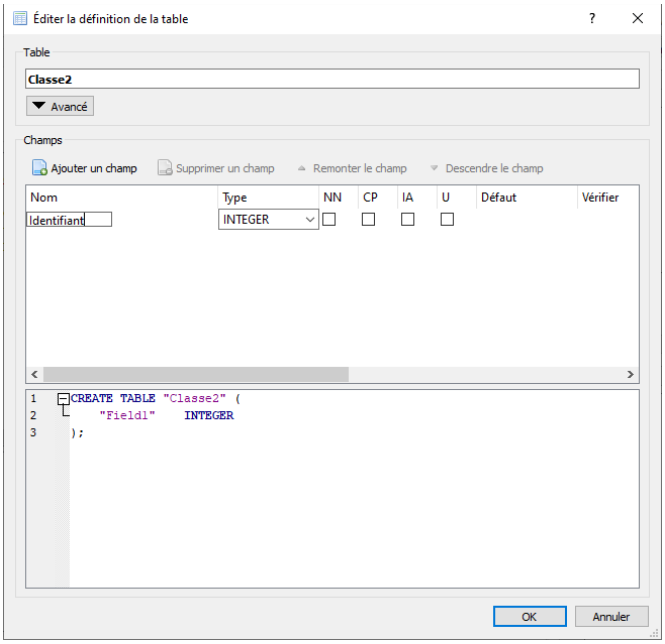
DB Browser for SQLite

La base de données s'appelle "Classe2"

On est en train de décrire l'attribut "Identifiant" (première colonne)

On observe en bas de la fenêtre le code généré automatiquement par le logiciel

DB Browser for SQLite



Dans la suite de ce document, nous étudierons les instructions SQL (et non l'utilisation de l'interface graphique).

Syntaxe : Pour créer une base de données avec **DB Browser for SQLite ,** il suffit d'utiliser la requête suivante

```
CREATE TABLE ma_base_de_données
```

Remarque : cette instruction peut différer légèrement selon le SGBD utilisé : on rencontre parfois :

```
CREATE DATABASE ma_base_de_données
```

```
Exemple : CREATE TABLE "Classe" (
     "Identifiant"  INTEGER PRIMARY KEY,
     "Nom"     TEXT,
     "Prenom"  TEXT,
     "Genre"   TEXT,
     "Age"     INTEGER,
     "Spe"     TEXT,
     "MoyenneGenerale"   REAL
);
```

Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre

Ci-dessus, on a créé une base de données comportant les attributs : "Identifiant" ; "Nom" ; "Prénom" ; "Genre" ; "Age" ; "Spe" et "MoyenneGenerale"
Remarque 1 : on précise le type de données correspondant à chaque attribut : texte ; nombre entier ; nombre réel...

Remarque 2 : on précise la ou les clés primaires après le type avec le mot clé PRIMARY KEY. Les clés étrangères sont précisées de la même façon avec le mot clé FOREIGN KEY.

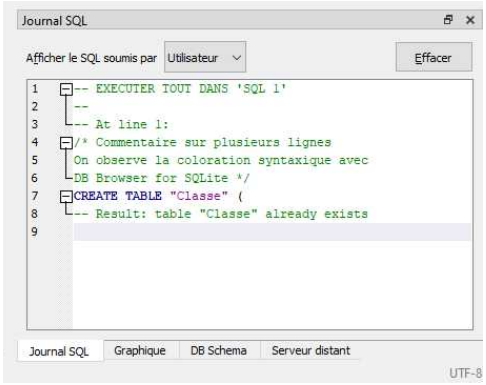
Remarque 2 : à l'issue de cette première instruction, on n'a définit que la structure de la base de données : on n'a fait aucun enregistrement ("remplir les lignes").

Remarque 3 : avec le logiciel **DB Browser for SQLite**, on bénéficie de la coloration syntaxique.

```
/* Commentaire sur plusieurs lignes
On observe la coloration syntaxique avec
DB Browser for SQLite */
CREATE TABLE "Classe" (   -- Commentaire sur une ligne : on utilise deux caractères "moins"
     "Identifiant"   INTEGER,  -- Attribut Identifiant : nombre entier
     "Nom"   TEXT,   -- Nom : texte ou chaîne de caractères
     "Prenom"    TEXT,
     "Genre" TEXT,
     "Age"   INTEGER,
     "Spe"   TEXT,
     "MoyenneGenerale"   REAL
);   -- On termine ces instructions par un point-virgule
```

Attention : il peut arriver par mégarde qu'une base de données portant le même nom existe déjà ! Dans ce cas, le SGBD générera un message d'erreur.

```
/* Commentaire sur plusieurs lignes
On observe la coloration syntaxique avec
DB Browser for SQLite */
CREATE TABLE "Classe" (   -- Commentaire sur une ligne : on utilise deux caractères "moins"
     "Identifiant"   INTEGER,  -- Attribut Identifiant : nombre entier
     "Nom"   TEXT,   -- Nom : texte ou chaîne de caractères
     "Prenom"    TEXT,
     "LangueVivante" TEXT,
     "MoyenneGenerale"   REAL
);   -- On termine ces instructions par un point-virgule
```



On obtient le message suivant :

-- Result: table "Classe" already exists

Pour éviter d'avoir cette erreur, il convient d'utiliser la requête suivante pour MySQL:

```
CREATE TABLE IF NOT EXISTS ma_base
```

L'option IF NOT EXISTS permet juste de ne pas retourner d'erreur si une base du même nom existe déjà.

La base de données ne sera pas écrasée.

Exemple :

```
CREATE TABLE "Classe" (
     "Identifiant"  INTEGER PRIMARY KEY,
     "Nom"     TEXT,
     "Prenom"  TEXT,
     "Genre"   TEXT,
     "Age"     INTEGER,
     "Spe"     TEXT,
     "MoyenneGenerale"   REAL
);
```

Exercice 1 : coder les instructions ci-dessus avec le logiciel **DB Browser for SQLite**

Exercice 2 :

Que font les instruction ci-dessous :

```
CREATE TABLE "ClasseT1" (
     "Identifiant"  INTEGER PRIMARY KEY,
     "Nom"     TEXT,
     "Prenom"  TEXT,
     "Genre"   TEXT,
     "Age"     INTEGER,
     "Spe"     TEXT,
     "LV1"     TEXT,
);
```

Exercice 3 :

Que font les instructions ci-dessous :

```
CREATE TABLE "NotesNSIT1" (
     "IdentifiantEleve"  INTEGER PRIMARY KEY,
     "DS1"     REAL,
     "DS2"     REAL,
     "DS3"     REAL,
     "QCM1"    INTEGER,
     "QCM2"    INTEGER,
     "DM1"     TEXT,
     "DM2"     TEXT,
     "DM3"     TEXT,
);
```

Exercice 4 :

Donner la requête en langage SQL pour créer la base de donnée ci-dessous :

IdEleve	DS1	DS2	DS3	QCM1	QCM2	DM1	DM2	DM3
Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre

Où :
"IdEleve" est une chaine de caractère, c'est également la clé primaire de la table ; "DS1" ; "DS2" et "DS3" sont des notes sur 20 (précises au demi-point) ; "QCM1" ; "QCM2" et "QCM3" sont des notes entières sur 10 et "DM1" ; "DM2" et "DM3" sont des notes sous forme de lettres.

Exercice 4 :

Donner la requête en langage SQL pour créer une base de données permettant de décrire les caractéristiques d'un parc de véhicules de location : numéro d'immatriculation ; marque ; modèle ; couleur ; année de fabrication ; type de carburant ; puissance moteur ; puissance fiscale et nombre total de kilomètres.

II.2/ Insersion d'éléments dans une base de donnée

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 entrée à la base existante ou plusieurs entrées en une seule requête. Les données correspondent à des entrées ou des enregistrements (tuple).

II.2.1/ Insertion d'une entrée

Pour insérer des données dans une base de donnée, il y a 2 possibilités :

- Insérer une entrée en indiquant toutes les informations pour chaque attribut existant (en respectant l'ordre)
- Insérer une entrée en spécifiant les attributs que l'on souhaite compléter. On peut ainsi insérer un enregistrement en renseignant uniquement une partie des attributs

a) Insérer une entrée en spécifiant tous les attributs

La syntaxe de la requête SQL pour remplir une seule ligne est la suivante :

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

Exemple de requête SQL :

Tout d'abord, il faudra avoir définit la base de donnée "Classe" :

```
CREATE TABLE "Classe" (
     "Identifiant"  INTEGER PRIMARY KEY,
     "Nom"     TEXT,
     "Prenom"  TEXT,
     "Genre"   TEXT,
     "Age"     INTEGER,
     "Spe"     TEXT,
     "MoyenneGenerale"   REAL
);
```

Puis pour insérer un élève, on fera :

```
INSERT INTO Classe VALUES
('93012720203101', 'Aime-Bappé', 'Kylian', 'M',15,'NSI',19.5);
```

Voici le résultat obtenu :

Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
93012720203101	Aime-Bappé	Kylian	M	15	NSI	19.5

On peut aussi décrire tous les attributs à renseigner comme suit :

```
INSERT INTO Classe
(Identifiant,Nom,Prenom,Genre,Age,Spe,MoyenneGenerale)
 VALUES   ('93012720203102', 'Bouhadie', 'Sarah', 'F',17,'LLCE',11.5);
```

Ce qui donne :

Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
93012720203101	Aime-Bappé	Kylian	M	15	NSI	19.5
93012720203102	Bouhadie	Sarah	F	17	LLCE	11.5

b) Insérer une entrée en spécifiant seulement quelques colonnes (attributs)

On n'est pas obligé de décrire tous les attributs d'un enregistrement : on peut en décrire qu'une partie.

Cette deuxième requête est très similaire à celle du dessus, mais il faut indiquer le nom des attributs avant "VALUES". La syntaxe est la suivante :

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

Remarque : il est possible de ne pas renseigner tous les attributs. De plus, l'ordre des attributs n'est pas important : on peut les permuter sans problème.

Exemple : suite à la requête :

```
INSERT INTO Classe (Identifiant,Nom,Prenom,Genre,Age)
 VALUES
 ('93012720203103', 'Casquecarrineau', 'Delphine', 'F',17);
```

on obtient :

Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
93012720203101	Aime-Bappé	Kylian	M	15	NSI	19.5
93012720203102	Bouhadie	Sarah	F	17	LLCE	11.5
93012720203103	Casquecarrineau	Delphine	F	17		

De même, avec :

```
INSERT INTO Classe (Identifiant,Age,Prenom,Genre,Spe,Nom)
 VALUES
 ('93012720203104', 17, 'Neymar', 'M','SVT','Dassilva');
```

on aura :

Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
93012720203101	Aime-Bappé	Kylian	M	15	NSI	19.5
93012720203102	Bouhadie	Sarah	F	17	LLCE	11.5
93012720203103	Casquecarrineau	Delphine	F	17		
93012720203104	Dassilva	Neymar	M	17	SVT	

Exercice : tester ces requêtes SQL dans **DB Browser for SQLite**

II.2.2/ Insertion de plusieurs lignes à la fois

Il est possible d'ajouter plusieurs entrées à une table avec une seule requête. Pour ce faire, il convient d'utiliser la syntaxe suivante :

```
INSERT INTO nom_table (nom_attribut_1, nom_attribut_2, ...)
VALUES
('valeur 11', 'valeur 12', ...)
('valeur 21', 'valeur 22', ...)
('valeur 31', 'valeur 32', ...)
('valeur 41', 'valeur 42', ...)
... ;
```

Exemple :

```
INSERT INTO Classe
(Identifiant,Nom,Prenom,Genre,Age,Spe,MoyenneGenerale)
 VALUES
 ('93012720203105', 'Deschant', 'Didier', 'M',21,'SI',9.5),
 ('93012720203106', 'Diannie', 'Kadidiatou', 'F',17,'Maths',11),
 ('93012720203107', 'Dillacre', 'Corine', 'F',17,'SPC',15),
 ('93012720203108', 'Emeboque', 'Griedge', 'F',16,'SVT',14.5),
 ('93012720203109', 'Grise-Manne', 'Antoine', 'M',17,'Maths',10),
 ('93012720203110', 'Henri', 'Amandine', 'F',17,'HLP',12.5);
```

Voici ce qu'on obtient une fois cette requête exécutée.

Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
93012720203101	Aime-Bappé	Kylian	M	15	NSI	19.5
93012720203102	Bouhadie	Sarah	F	17	LLCE	11.5
93012720203103	Casquecarrineau	Delphine	F	17		
93012720203104	Dassilva	Neymar	M	17	SVT	
93012720203105	Deschant	Didier	M	21	SI	9.5
93012720203106	Diannie	Kadidiatou	F	17	Maths	11.0
93012720203107	Dillacre	Corine	F	17	SPC	15.0
93012720203108	Emeboque	Griedge	F	16	SVT	14.5
93012720203109	Grise-Manne	Antoine	M	17	Maths	10.0
93012720203110	Henri	Amandine	F	17	HLP	12.5

Exercice : tester ces requêtes SQL dans **DB Browser for SQLite** et vérifier que votre base de données correspond au tableau ci-dessus.

II.2.3/ Insertion d'un attribut

On peut rajouter un attribut à notre base de données comme suit :

```
ALTER TABLE Classe ADD COLUMN "Mention" TEXT;
```

Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale	Mention
93012720203101	Aime-Bappé	Kylian	M	15	NSI	19.5	
93012720203102	Bouhadie	Sarah	F	17	LLCE	11.5	
93012720203103	Casquecarrineau	Delphine	F	17			
93012720203104	Dassilva	Neymar	M	17	SVT		
93012720203105	Deschant	Didier	M	21	SI	9.5	
93012720203106	Diannie	Kadidiatou	F	17	Maths	11.0	
93012720203107	Dillacre	Corine	F	17	SPC	15.0	
93012720203108	Emeboque	Griedge	F	16	SVT	14.5	
93012720203109	Grise-Manne	Antoine	M	17	Maths	10.0	
93012720203110	Henri	Amandine	F	17	HLP	12.5	

Exercice : écrire la requête SQL qui permet de rajouter à cette base de donnée un attribut "Admis" (dans laquelle on mettra les valeurs "oui" ou "non").

II.3/ Mise à jours des bases de données

La commande `UPDATE` permet de réaliser des modifications sur des entrées existantes.

Bien souvent cette commande est utilisée avec `WHERE` pour spécifier sur quelles lignes doivent porter la ou les modifications.

II.3.1/ Une mise à jours à la fois

La syntaxe basique d'une requête utilisant UPDATE est la suivante :

```
UPDATE nom_table
SET nom_attribut_1 = 'nouvelle valeur'
WHERE condition
```

Description : cette requête permet d'attribuer une nouvelle valeur à l'attribut nom_attribut_1 pour les entrées qui respectent la condition décrite avec `WHERE`.

Remarque : il est aussi possible d'attribuer la même valeur à l'attribut nom_attribut_1 pour toutes les entrées d'une table si la condition `WHERE` n'était pas utilisée.

Exemple 1 : on repart du tableau ci-dessous :

Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
93012720203101	Aime-Bappé	Kylian	M	15	NSI	19.5
93012720203102	Bouhadie	Sarah	F	17	LLCE	11.5
93012720203103	Casquecarrineau	Delphine	F	17		
93012720203104	Dassilva	Neymar	M	17	SVT	
93012720203105	Deschant	Didier	M	21	SI	9.5
93012720203106	Diannie	Kadidiatou	F	17	Maths	11.0
93012720203107	Dillacre	Corine	F	17	SPC	15.0
93012720203108	Emeboque	Griedge	F	16	SVT	14.5
93012720203109	Grise-Manne	Antoine	M	17	Maths	10.0
93012720203110	Henri	Amandine	F	17	HLP	12.5

On exécute la requête suivante :

```
UPDATE Classe
SET Spe = 'NSI'
WHERE Nom = 'Casquecarrineau'     /* Casquecarrineau (Delphine) a pris la spé NSI ! */
```

et voici ce qu'on obtient :

Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
93012720203101	Aime-Bappé	Kylian	M	15	NSI	19.5
93012720203102	Bouhadie	Sarah	F	17	LLCE	11.5
93012720203103	Casquecarrineau	Delphine	F	17	NSI	
93012720203104	Dassilva	Neymar	M	17	SVT	
93012720203105	Deschant	Didier	M	21	SI	9.5
93012720203106	Diannie	Kadidiatou	F	17	Maths	11.0
93012720203107	Dillacre	Corine	F	17	SPC	15.0
93012720203108	Emeboque	Griedge	F	16	SVT	14.5
93012720203109	Grise-Manne	Antoine	M	17	Maths	10.0
93012720203110	Henri	Amandine	F	17	HLP	12.5

Exemple 2 :

```
UPDATE Classe
SET Spe = 'NSI'
WHERE Nom = 'Henri'   /* Henri (Amandine) a changé de spé HLP -> NSI !
*/
```

Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
93012720203101	Aime-Bappé	Kylian	M	15	NSI	19.5
93012720203102	Bouhadie	Sarah	F	17	LLCE	11.5
93012720203103	Casquecarrineau	Delphine	F	17	NSI	
93012720203104	Dassilva	Neymar	M	17	SVT	
93012720203105	Deschant	Didier	M	21	SI	9.5
93012720203106	Diannie	Kadidiatou	F	17	Maths	11.0
93012720203107	Dillacre	Corine	F	17	SPC	15.0
93012720203108	Emeboque	Griedge	F	16	SVT	14.5
93012720203109	Grise-Manne	Antoine	M	17	Maths	10.0
93012720203110	Henri	Amandine	F	17	NSI	12.5

II.3.2/ Plusieurs mises à jours à la fois

Il est également possible de réaliser plusieurs mises à jour en une seule requête : pour cela, il est nécessaire de séparer les attributions de valeur par des virgules. La syntaxe correspondante est :

```
UPDATE nom_table
SET nom_attribut_1 = 'valeur 1', colonne_2 = 'valeur 2', colonne_3 =
'valeur 3'
WHERE condition
```

Exemple 1 :

```
UPDATE Classe
SET Spe = 'NSI' , MoyenneGenerale = 8.5
WHERE Nom = 'Dassilva'   /* Dassilva (Neymar) a changé de spé SVT ->
NSI et a 8,5 de moyenne */
```

La requête ci-dessus permet d'effectuer 2 modifications à la fois : on change la spé et la moyenne générale de l'élève Dassilva Neymar.

Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
93012720203101	Aime-Bappé	Kylian	M	15	NSI	19.5
93012720203102	Bouhadie	Sarah	F	17	LLCE	11.5
93012720203103	Casquecarrineau	Delphine	F	17	NSI	
93012720203104	Dassilva	Neymar	M	17	NSI	8.5
93012720203105	Deschant	Didier	M	21	SI	9.5
93012720203106	Diannie	Kadidiatou	F	17	Maths	11.0
93012720203107	Dillacre	Corine	F	17	SPC	15.0
93012720203108	Emeboque	Griedge	F	16	SVT	14.5
93012720203109	Grise-Manne	Antoine	M	17	Maths	10.0
93012720203110	Henri	Amandine	F	17	NSI	12.5

Exemple 2 :

```
UPDATE Classe
SET Spe = 'NSI'
WHERE Spe = 'Maths'   /* Tous les spés Maths on décidé de prendre
NSI ! */
```

Ici, on a réaffecté les élèves de la spé maths en la spé NSI !

Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
93012720203101	Aime-Bappé	Kylian	M	15	NSI	19.5
93012720203102	Bouhadie	Sarah	F	17	LLCE	11.5
93012720203103	Casquecarrineau	Delphine	F	17	NSI	
93012720203104	Dassilva	Neymar	M	17	NSI	8.5
93012720203105	Deschant	Didier	M	21	SI	9.5
93012720203106	Diannie	Kadidiatou	F	17	NSI	11.0
93012720203107	Dillacre	Corine	F	17	SPC	15.0
93012720203108	Emeboque	Griedge	F	16	SVT	14.5
93012720203109	Grise-Manne	Antoine	M	17	NSI	10.0
93012720203110	Henri	Amandine	F	17	NSI	12.5

Exemple 3 :

```
UPDATE Classe
SET MoyenneGenerale = 12
WHERE Identifiant = 93012720203103   /* La moyenne générale de
Casquecarrineau est de 12 */
```

Dans cet exemple, on a mis à jours la moyenne de Delphine Casquecarrineau : 12/20.

Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
93012720203101	Aime-Bappé	Kylian	M	15	NSI	19.5
93012720203102	Bouhadie	Sarah	F	17	LLCE	11.5
93012720203103	Casquecarrineau	Delphine	F	17	NSI	12.0
93012720203104	Dassilva	Neymar	M	17	NSI	8.5
93012720203105	Deschant	Didier	M	21	SI	9.5
93012720203106	Diannie	Kadidiatou	F	17	NSI	11.0
93012720203107	Dillacre	Corine	F	17	SPC	15.0
93012720203108	Emeboque	Griedge	F	16	SVT	14.5
93012720203109	Grise-Manne	Antoine	M	17	NSI	10.0
93012720203110	Henri	Amandine	F	17	NSI	12.5

Exemple 4 :

```
UPDATE Classe
SET Spe = NSI
WHERE Age >= 17 AND MoyenneGenerale > 14   /* Les élèves de plus de 17
ans et dont la moyenne est structement plus grande que 14 prennent la
spé NSI ! */
```

On a décidé dans cet exemple de réaffecter dans la spé NSI tous les élèves dont la moyenne générale est supérieure ou égale à 17/20

Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
93012720203101	Aime-Bappé	Kylian	M	15	NSI	19.5
93012720203102	Bouhadie	Sarah	F	17	LLCE	11.5
93012720203103	Casquecarrineau	Delphine	F	17	NSI	12.0
93012720203104	Dassilva	Neymar	M	17	NSI	8.5
93012720203105	Deschant	Didier	M	21	SI	9.5
93012720203106	Diannie	Kadidiatou	F	17	NSI	11.0
93012720203107	Dillacre	Corine	F	17	NSI	15.0
93012720203108	Emeboque	Griedge	F	16	SVT	14.5
93012720203109	Grise-Manne	Antoine	M	17	NSI	10.0
93012720203110	Henri	Amandine	F	17	NSI	12.5

Exercice 1 : On considère le tableau de l'exemple 4. Que fait la requête suivante ?

```
UPDATE Classe
SET MoyenneGenerale = 10.7
WHERE Prenom = "Didier"
```

Exercice 2 : On considère le tableau de l'exemple 4. Que fait la requête suivante ?

```
UPDATE Classe
SET Spe = "HG"
WHERE Spe = "NSI" AND MoyenneGenerale <= 10
```

Exercice 3 : On considère le tableau de l'exemple 4. Que fait la requête suivante ?

```
UPDATE Classe
SET MoyenneGenerale =  MoyenneGenerale+1
WHERE Spe = "NSI" AND MoyenneGenerale <= 10
```

Exercice 4 : On considère le tableau de l'exemple 4. Que font les deux requêtes ci-dessous ?

```
ALTER TABLE Notes ADD COLUMN "Mention" TEXT
ALTER TABLE Notes ADD COLUMN "Admis" TEXT;
```

Exercice 5 : Ecrire une requete SQL qui place dans la colonne "Mention" : "Admis" selon que la moyenne générale est supérieure ou non à 10.

Exercice 6 : Ecrire un ensemble de requetes SQL qui place dans la colonne "Mention" : "Admis TB" ; "Admis B" ; "Admis AB" ; "Admis" ; "Second Groupe" ou "Refusé" selon la moyenne générale.

II.4/ Suppression d'un base de données

La commande `DELETE` en SQL permet de supprimer des entrées dans une table.

En utilisant cette commande associée à `WHERE` il est possible de sélectionner les entrées concernées qui seront supprimées.

On peut également détruire toutes les entrées d'une table.

Attention : Avant d'essayer de supprimer des entrées, il est recommandé d'effectuer une sauvegarde de la base de données, ou tout du moins de la table concernée par la suppression. Ainsi, s'il y a une mauvaise manipulation il est toujours possible de restaurer les données.

La syntaxe pour supprimer des entrées est la suivante :

```
DELETE FROM `nom_table`
WHERE condition
```

Attention : s'il n'y a pas de condition WHERE alors **toutes** les entrées seront supprimées et la table sera alors vide.

II.4.1/ Supprimer une seule ligne

On repart de cet exemple

Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
93012720203101	Aime-Bappé	Kylian	M	15	NSI	19.5
93012720203102	Bouhadie	Sarah	F	17	LLCE	11.5
93012720203103	Casquecarrineau	Delphine	F	17		
93012720203104	Dassilva	Neymar	M	17	SVT	
93012720203105	Deschant	Didier	M	21	SI	9.5
93012720203106	Diannie	Kadidiatou	F	17	Maths	11.0
93012720203107	Dillacre	Corine	F	17	SPC	15.0
93012720203108	Emeboque	Griedge	F	16	SVT	14.5
93012720203109	Grise-Manne	Antoine	M	17	Maths	10.0
93012720203110	Henri	Amandine	F	17	HLP	12.5

On exécute la requête ci-dessous :

```
DELETE FROM Classe
WHERE Identifiant = 93012720203105
```

et on obtient :

Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
93012720203101	Aime-Bappé	Kylian	M	15	NSI	19.5
93012720203102	Bouhadie	Sarah	F	17	LLCE	11.5
93012720203103	Casquecarrineau	Delphine	F	17		
93012720203104	Dassilva	Neymar	M	17	SVT	
93012720203106	Diannie	Kadidiatou	F	17	Maths	11.0
93012720203107	Dillacre	Corine	F	17	SPC	15.0
93012720203108	Emeboque	Griedge	F	16	SVT	14.5
93012720203109	Grise-Manne	Antoine	M	17	Maths	10.0
93012720203110	Henri	Amandine	F	17	HLP	12.5

II.4.2/ Supprimer plusieurs lignes

Avec le langage SQL, on peut supprimer plusieurs entrées en mettant une condition adéquate.

Exemple :

```
DELETE FROM Classe
WHERE Identifiant >= 93012720203107
```

Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
93012720203101	Aime-Bappé	Kylian	M	15	NSI	19.5
93012720203102	Bouhadie	Sarah	F	17	LLCE	11.5
93012720203103	Casquecarrineau	Delphine	F	17		
93012720203104	Dassilva	Neymar	M	17	SVT	
93012720203106	Diannie	Kadidiatou	F	17	Maths	11.0

Ci-dessus, on a supprimé les élèves numéro : 93012720203107 ; 93012720203108 ; 93012720203109 et 93012720203110

En une seule requête, on a effacé 4 enregistrements.

Remarque : on peut faire des combinaisons logiques sur les conditions (cf. cours sur les opérateurs logiques).

Exemple :

```
DELETE FROM Classe
WHERE Identifiant >= 93012720203106 OR  Identifiant = 93012720203102
```

Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
93012720203101	Aime-Bappé	Kylian	M	15	NSI	19.5
93012720203103	Casquecarrineau	Delphine	F	17		
93012720203104	Dassilva	Neymar	M	17	SVT	

II.4.3/ Supprimer plusieurs lignes

On exécute tout simplement :

```
DELETE FROM Classe
```

Et voici ce qu'on obtient :

Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre

III/ Manipulation des données avec SQL

L'utilisation la plus courante de SQL consiste à lire / rechercher des données présentes dans une base de données.

Cela se réalise grâce à la commande `SELECT`, qui retourne des enregistrements dans un tableau de résultat. Cette commande peut sélectionner un ou plusieurs attributs d'une table.

III.1/ Afficher une sélection d'attributs avec SELECT

III.1.1/ Afficher un attribut

L'utilisation basique de cette commande s'effectue de la manière suivante:

```
SELECT nom_de_l_attribut FROM nom_de_la_table
```

Cette requête SQL va **sélectionner** (`SELECT`) l'attribut "nom_de_l_attribut" **provenant** (`FROM`) de la table appelé "nom_de_la_table".

Exemple :

```
SELECT Spe FROM Classe
```

Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
93012720203101	Aime-Bappé	Kylian	M	15	NSI	19.5
93012720203102	Bouhadie	Sarah	F	17	LLCE	11.5
93012720203103	Casquecarrineau	Delphine	F	17	NSI	12.0
93012720203104	Dassilva	Neymar	M	17	NSI	8.5
93012720203105	Deschant	Didier	M	21	SI	9.5
93012720203106	Diannie	Kadidiatou	F	17	NSI	11.0
93012720203107	Dillacre	Corine	F	17	NSI	15.0
93012720203108	Emeboque	Griedge	F	16	SVT	14.5
93012720203109	Grise-Manne	Antoine	M	17	NSI	10.0
93012720203110	Henri	Amandine	F	17	NSI	12.5

Voici le résultat affiché :

	Spe
1	NSI
2	LLCE
3	NSI
4	NSI
5	SI
6	NSI
7	NSI
8	SVT
9	NSI
10	NSI

III.1.2/ Obtenir plusieurs attributs avec SELECT

Avec la même table il est possible de lire plusieurs attributs à la fois.

Pour cela, on sépare les noms des attributs souhaités par une virgule.

Exemple :

```
SELECT Spe, MoyenneGenerale FROM Classe
```

Voici le résultat affiché :

	Spe	MoyenneGenerale
1	NSI	19.5
2	LLCE	11.5
3	NSI	12.0
4	NSI	8.5
5	SI	9.5
6	NSI	11.0
7	NSI	15.0
8	SVT	14.5
9	NSI	10.0
10	NSI	12.5

III.1.2/ Obtenir tous les attributs avec SELECT

Il est possible de retourner automatiquement tous les attributs d'une table sans avoir à connaître le nom de tous les attributs.

Au lieu de lister tous les attributs, il faut simplement utiliser le caractère "*" (étoile) : il permet de sélectionner tous les attributs et s'utilise de la manière suivante :

```
SELECT * FROM nom_de_la_table
```

Cette requête SQL retourne exactement les mêmes attributs qu'il y a dans la table.

Exemple :

```
SELECT * FROM Classe
```

Le requête ci-dessus donne :

	Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
1	93012720203101	Aime-Bappé	Kylian	M	15	NSI	19.5
2	93012720203102	Bouhadie	Sarah	F	17	LLCE	11.5
3	93012720203103	Casquecarrineau	Delphine	F	17	NSI	12.0
4	93012720203104	Dassilva	Neymar	M	17	NSI	8.5
5	93012720203105	Deschant	Didier	M	21	SI	9.5
6	93012720203106	Diannie	Kadidiatou	F	17	NSI	11.0
7	93012720203107	Dillacre	Corine	F	17	NSI	15.0
8	93012720203108	Emeboque	Griedge	F	16	SVT	14.5
9	93012720203109	Grise-Manne	Antoine	M	17	NSI	10.0
10	93012720203110	Henri	Amandine	F	17	NSI	12.5

III.2/ Afficher une sélection de lignes avec SELECT

Avec la requête `SELECT ... WHERE`, il est possible de faire un "filtrage"sur les entrées comme suit :

```
SELECT *
FROM nom_table
WHERE condition
```

On affiche tous les attributs (grâce à *) mais pas toutes les entrées à cause de la condition.

Exemple 1 :

La requête SQL : `SELECT * FROM Classe WHERE Spe = 'NSI'` donne :

	Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
1	93012720203101	Aime-Bappé	Kylian	M	15	NSI	19.5
2	93012720203103	Casquecarrineau	Delphine	F	17	NSI	12.0
3	93012720203104	Dassilva	Neymar	M	17	NSI	8.5
4	93012720203106	Diannie	Kadidiatou	F	17	NSI	11.0
5	93012720203107	Dillacre	Corine	F	17	NSI	15.0
6	93012720203109	Grise-Manne	Antoine	M	17	NSI	10.0
7	93012720203110	Henri	Amandine	F	17	NSI	12.5

Exemple 2 :

La requête SQL :

```
SELECT * FROM Classe WHERE Spe = 'NSI' AND MoyenneGenerale > 12
```

donne :

	Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
1	93012720203101	Aime-Bappé	Kylian	M	15	NSI	19.5
2	93012720203107	Dillacre	Corine	F	17	NSI	15.0
3	93012720203110	Henri	Amandine	F	17	NSI	12.5

III.3/ Opérateurs de comparaisons

On a vu ci-dessus, qu'on peut faire du filtrage sur les données en utilisant des mots clés comme AND ; OR ...
Il existe de nombreux opérateurs de comparaisons.

La liste ci-jointe présente quelques uns des opérateurs les plus couramment utilisés.

Opérateur	**Description**
=	Égale
<>	Pas égale
!=	Pas égale
>	Strictement Supérieur à
<	Strictement Inférieur à
>=	Supérieur ou égale à
<=	Inférieur ou égale à
IN	Liste de plusieurs valeurs possibles
BETWEEN	Valeur comprise dans un intervalle donnée (utile pour les nombres ou dates)
LIKE	Recherche en spécifiant le début, milieu ou fin d'un mot.
IS NULL	Valeur est nulle
IS NOT NULL	Valeur n'est pas nulle

Il est possible de combiner ces propositions par des opérateurs logiques.

OR	OU logique : vrai si et seulement si une des 2 propositions est vraie
AND	ET logique : vrai si et seulement si une les 2 propositions sont vraies
NOT	NON logique : vrai si et seulement si la proposition est fausse

III.3.1/ Opérateur logique OR

Syntaxe en SQL :

```
SELECT nom_attributs FROM nom_table
 WHERE condition1 OR condition2
```

Exemple :

```
SELECT * FROM Classe WHERE Spe = 'NSI' OR MoyenneGenerale > 12
```

III.3.2/ Opérateur logique AND

Syntaxe en SQL :

```
SELECT nom_attributs
FROM nom_table
  WHERE condition1 AND condition2
```

Exemple :

```
SELECT * FROM Classe WHERE Spe = 'NSI' AND MoyenneGenerale > 12
```

III.3.3/ Opérateur logique NOT

Syntaxe en SQL :

```
SELECT nom_attributs
FROM nom_table
  WHERE NOT condition
```

Exemple :

```
SELECT * FROM Classe WHERE NOT Spe = 'NSI'
```

Ce qui est équivalent à :

```
SELECT * FROM Classe WHERE Spe <> 'NSI'
```

Remarque : on veillera à utiliser des parenthèses : elles permettent d'améliorer la lecture et d'éviter les erreurs.

Exemple :

```
SELECT * FROM Classe WHERE NOT (Spe = 'NSI')
```

III.3.4/ Opérateur IN

L'opérateur logique `IN` dans SQL s'utilise avec la commande `WHERE` pour vérifier si un attribut est égale à une des valeurs comprise dans set de valeurs déterminés

C'est une méthode simple pour vérifier si les valeurs d'un attribut sont dans un ensemble de valeurs données.

Syntaxe

Pour chercher toutes les entrées où l'attribut "nom_attribut" est égale à 'valeur 1' OU 'valeur 2' ou 'valeur 3', il est possible d'utiliser la syntaxe suivante:

```
SELECT nom_attribut
FROM nom_table
WHERE nom_attribut IN ( valeur1, valeur2, valeur3, ... )
```

A savoir : entre les parenthèses il n'y a pas de limite du nombre d'arguments.

Cette syntaxe peut être associée à l'opérateur NOT pour recherche toutes les entrées qui ne sont pas égales à l'une des valeurs considérées.

Remarque : simplicité de l'opérateur `IN`

La syntaxe utilisée avec l'opérateur est plus simple que d'utiliser une succession d'opérateur `OR`. Pour le montrer concrètement avec un exemple, voici 2 requêtes qui retournerons les mêmes résultats, l'une utilise l'opérateur `IN`, tandis que l'autre utilise plusieurs `OR`.

Requête avec plusieurs `OR` :

```
SELECT Prenom
FROM Classe
WHERE prenom = 'Corine' OR prenom = 'Antoine' OR prenom = 'Amandine'
```

Requête équivalent avec l'opérateur `IN` :

```
SELECT Prenom
FROM Classe
WHERE prenom IN ( 'Corine', 'Antoine', 'Amandine' )
```

Exemple :

```
SELECT Prenom
FROM Classe
WHERE Prenom IN ( 'Corine', 'Antoine', 'Amandine' )
```

Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
93012720203101	Aime-Bappé	Kylian	M	15	NSI	19.5
93012720203102	Bouhadie	Sarah	F	17	LLCE	11.5
93012720203103	Casquecarrineau	Delphine	F	17	NSI	12.0
93012720203104	Dassilva	Neymar	M	17	NSI	8.5
93012720203105	Deschant	Didier	M	21	SI	9.5
93012720203106	Diannie	Kadidiatou	F	17	NSI	11.0
93012720203107	Dillacre	Corine	F	17	NSI	15.0
93012720203108	Emeboque	Griedge	F	16	SVT	14.5
93012720203109	Grise-Manne	Antoine	M	17	NSI	10.0
93012720203110	Henri	Amandine	F	17	NSI	12.5

	Prenom
1	Corine
2	Antoine
3	Amandine

Resultat

III.3.5/ Opérateur BETWEEN

L'opérateur `BETWEEN` est utilisé dans une requête SQL pour sélectionner un intervalle de données dans une requête utilisant `WHERE`. L'intervalle peut être constitué de chaînes de caractères, de nombres ou de dates. L'exemple le plus concret consiste par exemple à récupérer uniquement les enregistrements entre 2 nombres définis. Les bornes sont exclues de la sélection : c'est une comparaison au sens strict.

Syntaxe

L'utilisation de la commande `BETWEEN` s'effectue de la manière suivante :

```
SELECT *
FROM nom_table
WHERE nom_colonne BETWEEN 'valeur1' AND 'valeur2'
```

La requête suivante retournera toutes les entrées dont la valeur de l'attribut "nom_attribut" sera comprise entre **`valeur1`** et **`valeur2`**.

Exemple 1 :

```
SELECT Prenom, MoyenneGenerale
FROM Classe
WHERE MoyenneGenerale BETWEEN 12 AND 15
```

Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
93012720203101	Aime-Bappé	Kylian	M	15	NSI	19.5
93012720203102	Bouhadie	Sarah	F	17	LLCE	11.5
93012720203103	Casquecarrineau	Delphine	F	17	NSI	12.0
93012720203104	Dassilva	Neymar	M	17	NSI	8.5
93012720203105	Deschant	Didier	M	21	SI	9.5
93012720203106	Diannie	Kadidiatou	F	17	NSI	11.0
93012720203107	Dillacre	Corine	F	17	NSI	15.0
93012720203108	Emeboque	Griedge	F	16	SVT	14.5
93012720203109	Grise-Manne	Antoine	M	17	NSI	10.0
93012720203110	Henri	Amandine	F	17	NSI	12.5

	Prenom	MoyenneGenerale
1	Corine	15.0
2	Griedge	14.5
3	Amandine	12.5

Resultat

Remarque 1 : on observe bien que la comparaison est stricte : on ne sélectionne pas les lignes correspondant aux notes 12 ou 15.

Remarque 2 : il est également possible de comparer des chaines de caractères (ordre lexicographique).

Exemple 2 :

```
SELECT Prenom, Nom, MoyenneGenerale
FROM Classe
WHERE Nom BETWEEN 'A' AND 'Di'
```

Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
93012720203101	Aime-Bappé	Kylian	M	15	NSI	19.5
93012720203102	Bouhadie	Sarah	F	17	LLCE	11.5
93012720203103	Casquecarrineau	Delphine	F	17	NSI	12.0
93012720203104	Dassilva	Neymar	M	17	NSI	8.5
93012720203105	Deschant	Didier	M	21	SI	9.5
93012720203106	Diannie	Kadidiatou	F	17	NSI	11.0
93012720203107	Dillacre	Corine	F	17	NSI	15.0
93012720203108	Emeboque	Griedge	F	16	SVT	14.5
93012720203109	Grise-Manne	Antoine	M	17	NSI	10.0
93012720203110	Henri	Amandine	F	17	NSI	12.5

	Prenom	Nom	MoyenneGenerale
1	Kylian	Aime-Bappé	19.5
2	Sarah	Bouhadie	11.5
3	Delphine	Casquecarrineau	NULL
4	Neymar	Dassilva	NULL
5	Didier	Deschant	9.5

Resultat

Exercice 1 : afficher les noms et prénoms des élèves ayant exactement 17 ans

Exercice 2 : afficher les noms et prénoms des élèves n'ayant pas 17 ans

Exercice 3 : afficher les noms et prénoms des élèves ayant entre 15 (exclu) et 18 ans

III.3.6/ Opérateur LIKE

L'opérateur LIKE est utilisé dans la clause WHERE des requêtes SQL.

Ce mot-clé LIKE permet d'effectuer une recherche sur un modèle particulier.

Il est par exemple possible de rechercher les enregistrements dont la valeur d'un attribut commence par telle ou telle lettre.

Les modèles de recherches sont multiples.

Syntaxe : la syntaxe à utiliser pour utiliser l'opérateur LIKE est la suivante :

```
SELECT *
FROM nom_table
WHERE nom_attribut LIKE modele
```

Dans cet exemple le "modèle" peut correspondre à :

- `LIKE "%a"` : le caractère "%" est un caractère spécial qui remplace tous les autres caractères. Ainsi, ce modèle permet de rechercher toutes les chaines de caractère qui se termine par un "a" : par exemple "s**a**" ; "parl**a**" ; "programm**a**" ; "Alexandr**a**"
- `LIKE "a%"` : ce modèle permet de rechercher toutes les lignes de "colonne" qui commence par un "a" : par exemple "**a**menager" ; "**a**dn" ; "**a**basourdi " ; "**a**bordage"
- `LIKE "%a%"` : ce modèle est utilisé pour rechercher tous les enregistrement qui contiennent le caractère "a" : par exemple "**a**menager" ; "c**a**pter" ; "f**a**rine " ; "abord**a**ge"
- `LIKE "par%nt"` : ce modèle permet de rechercher les chaines qui commence par "par" et qui se terminent par "nt", comme "parent" ; "pareillement" ; "paressent" ; "parlement" ; "parlent" ...
- `LIKE "b_c"` : peu utilisé, le caractère "_" (underscore) peut être remplacé par n'importe quel caractère, mais un seul caractère uniquement (alors que le symbole pourcentage "%" peut être remplacé par plusieurs caractères. Ainsi, l'instruction `LIKE "b_c"` permet de retourner les lignes "**bac**" redouté par les bacheliers), "**bec**" ; "**bic**" (marque de stylo) ; **boc** (cabaret) ou encore **buc** (coque en navigation).

Exemple :

```
SELECT *
FROM Classe
WHERE Prenom LIKE "%i%"
```

Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
93012720203101	Aime-Bappé	Kylian	M	15	NSI	19.5
93012720203102	Bouhadie	Sarah	F	17	LLCE	11.5
93012720203103	Casquecarrineau	Delphine	F	17	NSI	12.0
93012720203104	Dassilva	Neymar	M	17	NSI	8.5
93012720203105	Deschant	Didier	M	21	SI	9.5
93012720203106	Diannie	Kadidiatou	F	17	NSI	11.0
93012720203107	Dillacre	Corine	F	17	NSI	15.0
93012720203108	Emeboque	Griedge	F	16	SVT	14.5
93012720203109	Grise-Manne	Antoine	M	17	NSI	10.0
93012720203110	Henri	Amandine	F	17	NSI	12.5

	Nom	Prenom	Spe
1	Aime-Bappé	Kylian	NSI
2	Casquecarrineau	Delphine	*NULL*
3	Deschant	Didier	SI
4	Diannie	Kadidiatou	Maths
5	Dillacre	Corine	SPC
6	Emeboque	Griedge	SVT
7	Grise-Manne	Antoine	Maths
8	Henri	Amandine	HLP

Resultat

Exercice 1 : que fait l'exemple ci-dessus ?

Exercice 2 : Ecrire une requête SQL qui affiche les notes des élèves dont le nom termine par "e".

III.4/ Afficher une sélection de lignes avec HAVING

L'instruction `HAVING` en SQL 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()`.

Syntaxe : l'utilisation de `HAVING` s'utilise de la manière suivante :

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

Cela permet donc de **sélectionner** les attributs de la table "nom_table" en **groupant** les entrées qui ont des valeurs identiques sur l'attribut "attribut1" et que la condition de `HAVING` soit respectée.

Exemple 1 : calcul de la moyenne en NSI et tri dans l'ordre croissant

```
CREATE TABLE IF NOT EXISTS "NotesNSI"
('NumeroDevoir','Nom','Prenom','Note');

INSERT INTO NotesNSI VALUES
 ('DS1', 'Aime-Bappé', 'Kylian', 11.5),
 ('DS1', 'Bouhadie', 'Sarah', 10),
 ('DS1', 'Casquecarrineau', 'Delphine', 14),
 ('DS1', 'Dassilva', 'Neymar',11),
 ('DS1', 'Deschant', 'Didier', 12),
 ('DS1', 'Diannie', 'Kadidiatou', 11),
 ('DS1', 'Dillacre', 'Corine', 16),
 ('DS1', 'Emeboque', 'Griedge', 9),
 ('DS1', 'Grise-Manne', 'Antoine',5),
 ('DS1', 'Henri', 'Amandine', 17);

INSERT INTO NotesNSI VALUES
 ('DS2', 'Aime-Bappé', 'Kylian', 12),
 ('DS2', 'Bouhadie', 'Sarah', 11),
 ('DS2', 'Casquecarrineau', 'Delphine', 12),
 ('DS2', 'Dassilva', 'Neymar',12),
 ('DS2', 'Deschant', 'Didier', 11),
 ('DS2', 'Diannie', 'Kadidiatou', 12),
 ('DS2', 'Dillacre', 'Corine', 15),
 ('DS2', 'Emeboque', 'Griedge', 10.5),
 ('DS2', 'Grise-Manne', 'Antoine',6),
 ('DS2', 'Henri', 'Amandine', 16.5);
```

NumeroDevoir	Nom	Prenom	Note
DS1	Aime-Bappé	Kylian	11.5
DS1	Bouhadie	Sarah	10
DS1	Casquecarrineau	Delphine	14
DS1	Dassilva	Neymar	11
DS1	Deschant	Didier	12
DS1	Diannie	Kadidiatou	11
DS1	Dillacre	Corine	16
DS1	Emeboque	Griedge	9
DS1	Grise-Manne	Antoine	5
DS1	Henri	Amandine	17
DS2	Aime-Bappé	Kylian	12
DS2	Bouhadie	Sarah	11
DS2	Casquecarrineau	Delphine	12
DS2	Dassilva	Neymar	12
DS2	Deschant	Didier	11
DS2	Diannie	Kadidiatou	12
DS2	Dillacre	Corine	15
DS2	Emeboque	Griedge	10.5
DS2	Grise-Manne	Antoine	6
DS2	Henri	Amandine	16.5

```
SELECT Nom, Prenom AS "Prénom", AVG(Note) AS "Moyennes NSI"
FROM  NotesNSI
GROUP BY Nom
ORDER BY Nom ASC
```

	Nom	Prénom	Moyennes NSI
1	Aime-Bappé	Kylian	11.75
2	Bouhadie	Sarah	10.5
3	Casquecarrineau	Delphine	13.0
4	Dassilva	Neymar	11.5
5	Deschant	Didier	11.5
6	Diannie	Kadidiatou	11.5
7	Dillacre	Corine	15.5
8	Emeboque	Griedge	9.75
9	Grise-Manne	Antoine	5.5
10	Henri	Amandine	16.75

Exemple 2 : calcul de la moyenne en NSI et filtrage des moyennes au dessus de 10

```
SELECT Nom, Prenom AS "Prénom", AVG(Note) AS "Moyennes NSI"
FROM  NotesNSI
GROUP BY Nom
HAVING AVG(Note) >= 10
ORDER BY Nom
```

	Nom	Prénom	Moyennes NSI
1	Aime-Bappé	Kylian	11.75
2	Bouhadie	Sarah	10.5
3	Casquecarrineau	Delphine	13.0
4	Dassilva	Neymar	11.5
5	Deschant	Didier	11.5
6	Diannie	Kadidiatou	11.5
7	Dillacre	Corine	15.5
8	Henri	Amandine	16.75

Exemple 3 : calcul de la moyenne en NSI et filtrage des moyennes au dessus de 10 et classement par ordre décroissant des notes et ordre croissant des noms

```
SELECT Nom, Prenom AS "Prénom", AVG(Note) AS "Moyennes NSI"
FROM  NotesNSI
GROUP BY Nom
HAVING AVG(Note) >= 10
ORDER BY AVG(Note) DESC , Nom ASC;
```

	Nom	Prénom	Moyennes NSI
1	Henri	Amandine	16.75
2	Dillacre	Corine	15.5
3	Casquecarrineau	Delphine	13.0
4	Aime-Bappé	Kylian	11.75
5	Dassilva	Neymar	11.5
6	Deschant	Didier	11.5
7	Diannie	Kadidiatou	11.5
8	Bouhadie	Sarah	10.5

III.5/ Opérateur DISTINCT

L'utilisation de la commande `SELECT` en SQL permet de lire toutes les données d'un ou plusieurs attributs. Cette commande peut potentiellement afficher des entrées en double. Pour éviter des redondances dans les résultats il faut simplement ajouter `DISTINCT` après le mot `SELECT`.

L'utilisation basique de cette commande consiste alors à effectuer la requête suivante:

```
SELECT DISTINCT nom_attribut
FROM nom_de_la_tableau
```

Cette requête sélectionne l'attribut "nom_attribut" de la table "nom_de_la_tableau" en évitant de retourner des doublons.

Exemple 1 :

```
SELECT Note
FROM  NotesNSI
ORDER BY Note DESC
```

	Note
1	17
2	16.5
3	16
4	15
5	14
6	12
7	12
8	12
9	12
10	12
11	11.5
12	11
13	11
14	11
15	11
16	10.5
17	10
18	9
19	6
20	5

Exemple 2 :

```
SELECT DISTINCT Note
FROM  NotesNSI
ORDER BY Note DESC
```

	Note
1	17
2	16.5
3	16
4	15
5	14
6	12
7	11.5
8	11
9	10.5
10	10
11	9
12	6
13	5

Exemple 3 :

```
SELECT avg(Note)
FROM  NotesNSI
GROUP BY Nom
ORDER BY avg(Note) DESC
```

	avg(Note)
1	16.75
2	15.5
3	13.0
4	11.75
5	11.5
6	11.5
7	11.5
8	10.5
9	9.75
10	5.5

Exemple 4 :

```
SELECT DISTINCT avg(Note)
FROM  NotesNSI
GROUP BY Nom
ORDER BY avg(Note) DESC
```

	avg(Note)
1	16.75
2	15.5
3	13.0
4	11.75
5	11.5
6	10.5
7	9.75
8	5.5

Exemple 5 :

```
SELECT DISTINCT Note, count(Note)
FROM  NotesNSI
GROUP BY Note
ORDER BY Note DESC
```

	Note	count(Note)
1	17	1
2	16.5	1
3	16	1
4	15	1
5	14	1
6	12	5
7	11.5	1
8	11	4
9	10.5	1
10	10	1
11	9	1
12	6	1
13	5	1

III.6/ Opérateur INNER JOIN

Dans le langage SQL la commande `INNER JOIN`, est un type de jointures très communes pour lier plusieurs tables entre-elles. Cette commande retourne les enregistrements lorsqu'il y a au moins une entée dans chaque attribut qui correspond à la condition.

Syntaxe : pour utiliser ce type de jointure il convient d'utiliser une requête SQL avec cette syntaxe :

```
SELECT *
FROM nom_table1
INNER JOIN nom_table2 ON nom_table1.id = nom_table2.fk_id
```

La syntaxe ci-dessus stipule qu'il faut sélectionner les enregistrements des tables **nom_table1** et **nom_table2** lorsque les données de l'attribut "id" de nom_table1 est égal aux données de l'attribut fk_id de nom_table2.

La jointure SQL peux aussi être écrite de la façon suivante :

```
SELECT *
FROM nom_table1
INNER JOIN nom_table2
WHERE nom_table1.id = nom_table2.fk_id
```

La syntaxe avec la condition `WHERE` est une manière alternative de faire la jointure mais qui possède l'inconvénient d'être moins facile à lire s'il y a déjà plusieurs conditions dans le `WHERE`.

Exemple :

Identifiant	Nom	Prenom	Genre	Age	Spe	MoyenneGenerale
93012720203101	Aime-Bappé	Kylian	M	15	NSI	19.5
93012720203102	Bouhadie	Sarah	F	17	NSI	11.5
93012720203103	Casquecarrineau	Delphine	F	17	NSI	12.0
93012720203104	Dassilva	Neymar	M	17	NSI	8.5
93012720203105	Deschant	Didier	M	21	NSI	9.5
93012720203106	Diannie	Kadidiatou	F	17	NSI	11.0
93012720203107	Dillacre	Corine	F	17	NSI	15.0
93012720203108	Emeboque	Griedge	F	16	NSI	14.5
93012720203109	Grise-Manne	Antoine	M	17	NSI	10.0
93012720203110	Henri	Amandine	F	17	NSI	12.5

Tableau : Classe

NumeroDevoir	Nom	Prenom	Note
DS1	Aime-Bappé	Kylian	11.5
DS1	Bouhadie	Sarah	10
DS1	Casquecarrineau	Delphine	14
DS1	Dassilva	Neymar	11
DS1	Deschant	Didier	12
DS1	Diannie	Kadidiatou	11
DS1	Dillacre	Corine	16
DS1	Emeboque	Griedge	9
DS1	Grise-Manne	Antoine	5
DS1	Henri	Amandine	17
DS2	Aime-Bappé	Kylian	12
DS2	Bouhadie	Sarah	11
DS2	Casquecarrineau	Delphine	12
DS2	Dassilva	Neymar	12
DS2	Deschant	Didier	11
DS2	Diannie	Kadidiatou	12
DS2	Dillacre	Corine	15
DS2	Emeboque	Griedge	10.5
DS2	Grise-Manne	Antoine	6
DS2	Henri	Amandine	16.5

Tableau : NotesNSI

```
SELECT MoyenneGenerale, Note
FROM Classe
INNER JOIN NotesNSI
ON Classe.Nom = NotesNSI.Nom
ORDER BY MoyenneGenerale ASC;
```

	MoyenneGenerale	Note
1	8.5	11
2	8.5	12
3	9.5	11
4	9.5	12
5	10.0	5
6	10.0	6
7	11.0	11
8	11.0	12
9	11.5	10
10	11.5	11
11	12.0	12
12	12.0	14
13	12.5	16.5
14	12.5	17
15	14.5	9
16	14.5	10.5
17	15.0	15
18	15.0	16
19	19.5	11.5
20	19.5	12

Ce résultat permet par exemple de voir s'il y a un lien entre la moyenne générale et les notes en NSI.

IV/ Exemple d'activités / projets

IV.1/ Activité 1 : création d'une base de données

1°) Donner les instructions SQL pour contruire cette table

Identifiant	Nom	Prenom	Genre	Age	Spe1	Spe2	LV1	LV2	MoyenneGenerale
Filtre	Filtre	Filtre	...				...	Fil...	Filtre

2°) Donner les instructions SQL pour insérer les 20 élèves de la classe comme suit :

Identifiant	Nom	Prenom	Genre	Age	Spe1	Spe2	LV1	LV2	MoyenneGenerale
93012720203101	Aime-Bappé	Kylian	M	15	NSI	Maths	Anglais	Espagnol	19.5
93012720203102	Bouhadie	Sarah	F	17	LLCE	SPC	Anglais	Italien	11.5
93012720203103	Casquecarrineau	Delphine	F	17	LCA	LLCE	Anglais	Espagnol	12.5
93012720203104	Dassilva	Neymar	M	17	SVT	HG	Anglais	Russe	13.0
93012720203105	Deschant	Didier	M	21	SI	SVT	Anglais	Espagnol	9.5
93012720203106	Diannie	Kadidiatou	F	17	Maths	SES	Anglais	Espagnol	11.0
93012720203107	Dillacre	Corine	F	17	SPC	NSI	Anglais	Allemand	15.0
93012720203108	Emeboque	Griedge	F	16	SVT	SPC	Anglais	Espagnol	14.5
93012720203109	Grise-Manne	Antoine	M	17	Maths	NSI	Anglais	Espagnol	10.0
93012720203110	Henri	Amandine	F	17	HLP	SES	Anglais	Russe	12.5
93012720203111	Henri	Thierry	M	19	SI	LCA	Anglais	Espagnol	10.0
93012720203113	Lessomez	Eugénie	F	17	HLP	NSI	Anglais	Espagnol	15.0
93012720203114	Lorrisse	Hugo	F	18	NSI	SES	Anglais	Allemand	11.0
93012720203115	Magerie	Adel	F	17	SES	NSI	Anglais	Italien	14.0
93012720203112	Maissy	Lionel	M	17	LLCE	SPC	Anglais	Espagnol	8.5
93012720203116	Poguebat	Paul	M	17	HLP	LLCE	Anglais	Chinois	12.0
93012720203117	Reinehard	Wendie	F	16	Maths	SES	Anglais	Espagnol	17.0
93012720203118	Ronalledeau	Kristiano	M	19	HLP	Maths	Anglais	Italien	11.0
93012720203119	Vieyras	Patrick	M	20	LLCE	SPC	Anglais	Allemand	9.5
93012720203120	Zyde-Anne	Zinedine	M	19	NSI	SES	Anglais	Espagnol	14.5

3°) Sélection : donner la requête SQL qui permet d'afficher :
- la liste des élèves qui ont ont choisi la spécialité NSI en premier choix
- la liste des élèves qui ont ont choisi la spécialité NSI
- la liste des élèves qui ont n'ont choisi la spécialité NSI
- la liste des élèves qui ont choisi la spécialité NSI et l'Espagnol
- la liste des élèves qui ont la moyenne
- la liste des filles qui ont la moyenne et ont choisi la spé NSI
- la moyenne de la classe
- la moyenne des élèves faisant spé NSI
- la moyenne des garçons ne faisant pas spé NSI

Solution de l'activité 1

1°)

```
CREATE TABLE "ClasseT1" (
     "Identifiant"  INTEGER PRIMARY KEY,
     "Nom"     TEXT,
     "Prenom"  TEXT,
     "Genre"   TEXT,
     "Age"     INTEGER,
     "Spe1"    TEXT,
     "Spe2"    TEXT,
     "LV1"     TEXT,
     "LV2"     TEXT,
     "MoyenneGenerale"   REAL
);
```

2°)

```
INSERT INTO ClasseT1
(Identifiant,Nom,Prenom,Genre,Age,Spe1,Spe2,LV1,LV2,MoyenneGenerale)
 VALUES
 ('93012720203101', 'Aime-Bappé', 'Kylian',
'M',15,'NSI','Maths','Anglais','Espagnol',19.5),
 ('93012720203102', 'Bouhadie', 'Sarah',
'F',17,'LLCE','SPC','Anglais','Italien',11.5),
 ('93012720203103', 'Casquecarrineau', 'Delphine',
'F',17,'LCA','LLCE','Anglais','Espagnol',12.5),
 ('93012720203104', 'Dassilva', 'Neymar',
'M',17,'SVT','HG','Anglais','Russe',13),
 ('93012720203105', 'Deschant', 'Didier',
'M',21,'SI','SVT','Anglais','Espagnol',9.5),
 ('93012720203106', 'Diannie', 'Kadidiatou',
'F',17,'Maths','SES','Anglais','Espagnol',11),
 ('93012720203107', 'Dillacre', 'Corine',
'F',17,'SPC','NSI','Anglais','Allemand',15),
 ('93012720203108', 'Emeboque', 'Griedge',
'F',16,'SVT','SPC','Anglais','Espagnol',14.5),
 ('93012720203109', 'Grise-Manne', 'Antoine',
'M',17,'Maths','NSI','Anglais','Espagnol',10),
 ('93012720203110', 'Henri', 'Amandine',
'F',17,'HLP','SES','Anglais','Russe',12.5),
 ('93012720203111', 'Henri', 'Thierry',
'M',19,'SI','LCA','Anglais','Espagnol',10),
 ('93012720203112', 'Maissy', 'Lionel',
'M',17,'LLCE','SPC','Anglais','Espagnol',8.5),
 ('93012720203113', 'Lessomez', 'Eugénie',
'F',17,'HLP','NSI','Anglais','Espagnol',15),
 ('93012720203114', 'Lorrisse', 'Hugo',
'F',18,'NSI','SES','Anglais','Allemand',11),
 ('93012720203115', 'Magerie', 'Adel',
'F',17,'SES','NSI','Anglais','Italien',14),
 ('93012720203116', 'Poguebat', 'Paul',
'M',17,'HLP','LLCE','Anglais','Chinois',12),
 ('93012720203117', 'Reinehard', 'Wendie',
```

```
'F',16,'Maths','SES','Anglais','Espagnol',17),
 ('93012720203118', 'Ronalledeau', 'Kristiano',
'M',19,'HLP','Maths','Anglais','Italien',11),
 ('93012720203119', 'Vieyras', 'Patrick',
'M',20,'LLCE','SPC','Anglais','Allemand',9.5),
 ('93012720203120', 'Zyde-Anne', 'Zinedine',
'M',19,'NSI','SES','Anglais','Espagnol',14.5);
```

3°)

```
SELECT Nom, Prenom
FROM ClasseTl
WHERE Spe1 = 'NSI';

SELECT Nom, Prenom
FROM ClasseTl
WHERE Spe1 = 'NSI' OR Spe2 = 'NSI';

SELECT Nom, Prenom
FROM ClasseTl
WHERE Spe1 <> 'NSI' AND Spe2 <> 'NSI';

SELECT Nom, Prenom
FROM ClasseTl
WHERE (Spe1 = 'NSI' OR Spe2 = 'NSI') AND (LV1 = 'Espagnol' OR LV2 =
'Espagnol');

SELECT Nom, Prenom
FROM ClasseTl
WHERE MoyenneGenerale >= 10;

SELECT Nom, Prenom
FROM ClasseTl
WHERE Genre = 'F' AND MoyenneGenerale >= 10 AND (Spe1 = 'NSI' OR Spe2
= 'NSI');

SELECT AVG(MoyenneGenerale) AS 'MoyenneClasse'
FROM ClasseTl;

SELECT AVG(MoyenneGenerale) AS 'MoyenneElevesNSI'
FROM ClasseTl
WHERE Spe1 = 'NSI' OR Spe2 = 'NSI';

SELECT AVG(MoyenneGenerale)AS 'MoyenneGarconsHorsNSI'
FROM ClasseTl
WHERE Genre = 'G' AND Spe1 <> 'NSI' AND Spe2 <> 'NSI';
```

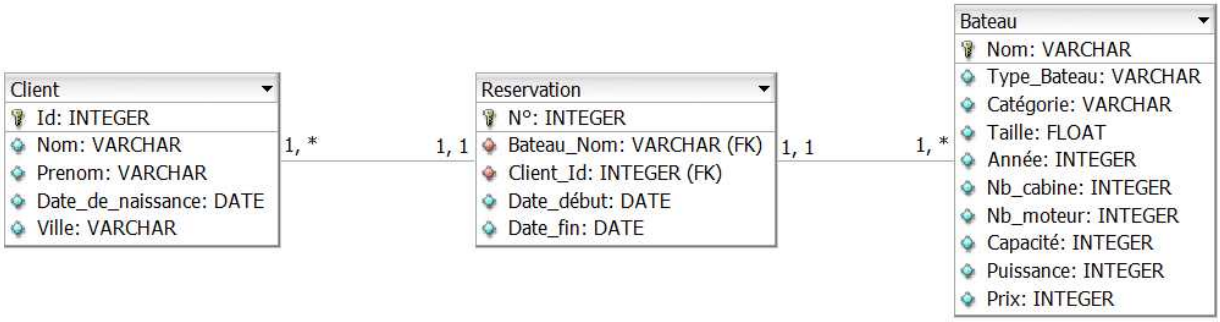
IV.2/ Activité 2 : *Gestion entreprise location de bateaux*

Remarque : Cette activité démarre avec une base de données dont la conception et le schéma relationnel est déjà fait. La conception de ceux-ci peut se faire avec les élèves plus tôt dans le chapitre au moment de l'étude du modèle relationnel. Il s'agirait alors de réfléchir aux relations, aux attributs et aux différentes clés primaires et étrangères intervennant dans la situation. L'élève dispose d'une base de données déjà remplies pour accelérer le processus de création.

La société de location de bateau Bateau Loco dispose d'une base de données pour gérer sa flotte de bateaux, ses dossiers clients et les réservations de bateaux effectuées.

Partie 1 : Création de la base de données

Voici le schéma relationnel de cette base de donnée :



Elle est composé de trois tables :

- La table **`Client`** dont la clé primaire est un nombre entier `INTEGER` nommé `Id` et est composé des attributs `Nom`, `Prenom` de type chaîne de caractères `VARCHAR`, `Date_de_naissance` de type `DATE` et `Ville` de type chaîne de caractère `VARCHAR` correspondant à la ville de résidence du client.
- La table **`Bateau`** dont la clé primaire est le nom du bateau `Nom` de type chaîne de charactère `VARCHAR` et est composé des attributs suivants permettant de décrire les caracteristiques du bateau :
  - son type `Type_Bateau` de type chaîne de caractères `VARCHAR` pouvant prendre les valeurs `'Bateau à moteur'` ou `'Voilier'` ;
  - la catégorie du bateau Catégorie de type chaîne de caractères VARCHAR pouvant prendre les valeurs `'Vedette'`, `'Multicoque'`, `'Voilier Plaisance'`, `'Voilier Course'`, `'Grand Voilier'`, `'Coque ouverte'` ou `'Yacht'` ;
  - la longueur du bateau `Taille` de type nombre flottant `FLOAT` ;
  - l'année de mise à l'eau du bateau `Année` de type nombre entier `INTEGER` ;
  - le nombre de cabine du bateau `Nb_cabine` de type nombre entier `INTEGER` ;
  - le nombre de moteur sur le bateau `Nb_moteur` de type nombre entier `INTEGER` ;
  - le nombre de personne maximum pouvant monter simultanement sur le bateau `Capacité` de type nombre entier `INTEGER` ;
  - la puissance en cheveaux du bateau `Puissance` de type nombre entier `INTEGER` ;
  - le prix de location journalier du bateau `Prix` de type nombre entier `INTEGER`.
- La table **`Reservation`** permettant d'enregistrer les réservations de bateaux des clients. La clé primaire est un nombre entier `INTEGER` correspondant au numéro de réservation et nommée `N°`. Une réservation est composé de deux clés étrangères: `Bateau_Nom`, clé primaire de la table `Bateau` et `Client_Id`, clé primaire de la table `Client`. Une réservation est également caractérisée par deux dates de type `DATE` : la date de début de réservation `Date_début` et la date de fin `Date_fin`.

Question 1 : Donner les instructions SQL permettant de créer ces trois tables.

La société dispose de la flotte suivante :

Nom	Type	Catégorie	Taille	Année	Nb_cabine	Nb_moteur	Capacité	Puissance	Prix
En-vau	Bateau à moteur	Yacht	23	2015	2	2	4	300	2800
La Mounine	Voilier	Voilier Plaisance	8,99	2014	2	1	6	21	165
Le Devenson	Bateau à moteur	Coque ouverte	4,4	2014	0	1	4	6	160
L'Eissadon	Bateau à moteur	Vedette	10,57	2017	1	2	4	300	700
Les Queyrons	Voilier	Voilier Course	9,2	2003	2	1	3	50	500
L'Escu	Voilier	Voilier Plaisance	11,44	2020	2	1	6	30	2300
L'Œil de verre	Bateau à moteur	Coque ouverte	6,15	2020	0	1	6	140	300
L'Oule	Bateau à moteur	Vedette	8,37	2015	1	2	2	200	650
Marseilleveyre	Voilier	Voilier Course	13,2	2003	2	1	5	50	1200
Morgiou	Voilier	Multicoque	18,9	2016	4	2	10	150	3000
Podestat	Voilier	Voilier Course	14,5	2010	4	1	10	110	1600
Port-miou	Bateau à moteur	Bateau de course	6,05	2017	0	1	7	115	249
Port-pin	Bateau à moteur	Yacht	22	2009	3	2	6	400	3500
Sormiou	Voilier	Grand Voilier	32,59	2011	4	2	10	440	12000
Sugiton	Voilier	Multicoque	11,55	2008	4	2	10	29	2000

Question 2 : Donner l'instruction SQL qui a permis d'ajouter le bateau Morgiou.

Question 3 : La société investie dans un second moteur pour rendre plus puissant le bateau Le Devenson faisant passer sa puissance à 20 cheveaux. Écrire l'instruction SQL permettant l'information de cet achat dans la base de donnée.

La base client de Bateau Loco est la suivante :

Id	Nom	Prenom	Date_de_naissance	Ville
1	Turing	Alan	23/06/1912	Londres
2	Lovelace	Ada	10/12/1815	Londres
3	Hopper	Grace	09/12/1906	New York
4	Kahn	Gilles	17/04/1946	Paris
5	Vaughan	Dorothy	20/09/1910	Kansas City
6	Boole	George	02/11/1815	Lincoln
7	Shannon	Claude	30/04/1916	Petoskey
8	Babbage	Charles	26/12/1791	Londres
9	Von Neumann	John	28/12/1903	Budapest
10	Moore	Gordon	03/01/1929	San Fransisco
11	Lamarr	Hedy	09/11/1914	Vienne
12	Hinton	Geoffrey	06/12/1947	Wimbledon

Question 4 : John Von Neumann a déménagé à Washington. Mettre à jour sa fiche dans la table client à l'aide d'une instuction SQL.

Les réservations de bateaux effectuées par les clients sont les suivantes :

N°	Bateau_Nom	Client_Id	Date_début	Date_fin
1	L'Escu	1	02/01/2020	05/01/2020
2	La Mounine	2	02/01/2020	03/01/2020
3	L'Oule	3	06/01/2020	10/01/2020
4	En-vau	4	08/01/2020	08/01/2020
5	Les Queyrons	5	10/01/2020	12/01/2020
6	L'Escu	1	13/01/2020	16/01/2020
7	Morgiou	6	25/01/2020	26/01/2020
8	La Mounine	7	27/01/2020	01/02/2020
9	L'Œil de Verre	8	27/01/2020	28/01/2020
10	Sormiou	9	29/01/2020	12/02/2020
11	Podestat	3	31/01/2020	05/02/2020
12	L'Oule	10	03/02/2020	07/02/2020
13	L'Eissadon	11	05/02/2020	07/02/2020
14	Port-pin	1	10/02/2020	03/03/2020
15	En-vau	6	12/02/2020	15/02/2020
16	Sugiton	12	14/02/2020	16/02/2020
17	Port-miou	3	24/02/2020	28/02/2020
18	Marseilleveyre	5	02/03/2020	06/03/2020

Question 5 : Hedy Lamarr réserve le Podestat entre le 04/03/2020 et le 06/03/2020. Mettre à jour la table des réservations.

Partie 2 : Requêtes sur la base de données

Pour chacune des situations suivantes, écrire une requête SQL permettant d'obtenir les informations voulues.

Question 1 : Un client souhaite obtenir la liste de tous les bateaux accompagnée de leurs caracteristiques.

Question 2 : Le patron de la société souhaite obtenir la liste des prenoms et noms de tous ses clients.

Question 3 : Le patron souhaite obtenir la liste de tous ses clients londonniens.

Question 4 : Un client souhaite réserver un voilier de course. Quels sont les bateaux correspondant à sa demande ?

Question 5 : Un client souhaite connaître les différentes puissance des voiliers multicoque.

Question 6 : Le patron souhaite connaître les villes de résidence de ces clients. Ne pas afficher de doublons.

Question 7 : Un employé souhaite connaître les dates auquelles a été réservé le bateau baptisé En-vau.

Question 8 : Un client souhaite connaître la liste des bateaux à moteur dont la puissance n'excède pas 200 cheveaux.

Question 9 : Un client participe à une course nautique de voilier. Il dispose d'un budget de 1 500 € par jour. Quels sont les bateaux accessible pour un tel budget ?

Question 10 : Un employé souhaite consulter les bateaux réservés par Alan Turing.

Question 11 : Un client souhaite louer un bateau à moteur pour sa famille composée de sa femme, ses deux enfants et lui. Il souhaite au minimum deux cabines. Quels bateaux la société peut lui proposer ?

Question 12 : Le patron souhaite connaître le prix moyen de ses yacht.

Question 13 : Pour une brochure publicitaire, la directrice de la communication souhaite afficher la liste des noms voiliers accompagnés de leur prix journalier et de l'année de vise à l'eau en les rangeant par ordre de prix décroissant.

Question 14 : Le bateau nommé En-vau a été dégradé. L'assurance souhaiterait connaître les noms et prénoms des clients l'ayant loué en les affichant dans l'ordre décroissant des dates de retour.

Corriger de l'activité :

Partie 1 : Création de la base de données

Question 1 :

```
CREATE TABLE "Client" (
      "Id"                INTEGER PRIMARY KEY,
      "Nom"               VARCHAR,
      "Prenom"            VARCHAR,
      "Date_de_naissance" DATE,
      "Ville"             VARCHAR
);

CREATE TABLE "Bateau" (
      "Nom"         VARCHAR PRIMARY KEY,
      "Type_Bateau" VARCHAR,
      "Catégorie"   VARCHAR,
      "Taille"      FLOAT,
      "Année"       INTEGER,
      "Nb_cabine"   INTEGER,
      "Nb_moteur"   INTEGER,
      "Capacité"    INTEGER,
      "Puissance"   INTEGER,
      "Prix"        INTEGER
);

CREATE TABLE "Reservation" (
      "N°"         INTEGER PRIMARY KEY,
      "Bateau_Nom" VARCHAR FOREIGN KEY REFERENCES Bateau(Nom),
      "Client_Id"  INTEGER FOREIGN KEY REFERENCES Client(Id),
      "Date_début" DATE,
      "Date_fin"   DATE
);
```

Question 2 :

```
INSERT INTO Bateau VALUES
('Morgiou', 'Voilier', 'Multicoque', '18,9', '2016', '4', '2', '10', '150', '3000');
```

Question 3 :

```
UPDATE Bateau
SET Nb_moteur = '2', Puissance = '20'
WHERE Nom='Morgiou';
```

Question 4 :

```
UPDATE Client
SET Ville = 'Washington'
WHERE Id=9;
```

Question 5 :

```
INSERT INTO Reservation (Bateau_Nom, Client_Id, Date_début, Date_fin) VALUES
("Podestat", 11, "04/03/2020", "06/03/2020");
```

Partie 2 : Requêtes sur la base de données

Question 1 :

```
SELECT *
FROM Bateau;
```

Question 2 :

```
SELECT Prenom, Nom
FROM Client;
```

Question 3 :

```
SELECT Prenom, Nom
FROM Client
WHERE Ville = 'Londres';
```

Question 4 :
```
SELECT *
FROM Bateau
WHERE Catégorie = 'Voilier course';
```

Question 5 :
```
SELECT Puissance
FROM Bateau
WHERE Catégorie = 'Multicoque';
```

Question 6 :
```
SELECT DISTINCT Ville
FROM Client;
```

Question 7 :
```
SELECT Date_début
FROM Reservation
WHERE Bateau_nom = 'En-vau';
```

Question 8 :
```
SELECT *
FROM Bateau
WHERE Type = 'Bateau à moteur' AND Puissance<=200;
```

Question 9 :
```
SELECT *
FROM Bateau
WHERE Catégorie = 'Voilier course'  AND Prix <= 1500;
```

Question 10 :
```
SELECT DISTINCT Bateau_nom
FROM Reservation
INNER JOIN Client ON Client.Id = Reservation.Client_Id
WHERE Client.Nom = 'Turing';
```

Question 11 :
```
SELECT *
FROM Bateau
WHERE Type = 'Bateau à moteur' AND Nb_cabine >= 2 AND Capacité >= 4;
```

Question 12 :
```
SELECT AVG(Prix)
FROM Bateau
WHERE Catégorie = 'Yacht';
```

Question 13 :
```
SELECT Nom, Prix, Année
FROM Bateau
WHERE Type = 'Voilier'
ORDER BY Prix DESC;
```

Question 14 :
```
SELECT Nom, Prenom
FROM CLIENT
INNER JOIN Reservation ON Client.Id = Reservation.Client_Id
WHERE Reservation.Bateau_Nom = 'En-vau'
ORDER BY Reservation.Date_fin DESC;
```

V/ Bibilographie

https://lipn.univ-paris13.fr/~boufares/ ***(un grand merci à M. Boufarès pour son cours sur les bases de données !)***

http://www.i3s.unice.fr/~edemaria/cours/c4.pdf

https://sql.sh/cours/create-database

https://www.sqlitetutorial.net

https://fr.wikipedia.org/wiki/Structured_Query_Language

https://fr.wikipedia.org/wiki/Système_de_gestion_de_base_de_données

https://openclassrooms.com/fr/courses/993975-apprenez-a-programmer-en-vb-net/992711-introduction-au-langage-sql

https://www.culture-informatique.net/cest-quoi-sql/

https://fr.wikibooks.org/wiki/Programmation_SQL/Langage_de_manipulation_de_données

https://sqlitebrowser.org/

https://pixees.fr/informatiquelycee/n_site/nsi_term_bd_sql.html

https://www.youboat.com/fr/