Bases de Données

Exercice 1. :

L'énoncé de cet exercice utilise les mots du langage SQL suivants :

`SELECT FROM, WHERE, JOIN ON, INSERT INTO VALUES, UPDATE, SET, DELETE, COUNT, AND, OR.`

Pour la gestion des réservations clients, on dispose d'une base de données nommée « gare » dont le schéma relationnel est le suivant :

`Train (numT, provenance, destination, horaireArrivee, horaireDepart)`

`Reservation (numR, nomClient, prenomClient, prix, #numT)`

Les attributs soulignés sont des clés primaires. L'attribut précédé de `#` est une clé étrangère.

La clé étrangère `Reservation.numT` fait référence à la clé primaire `Train.numT`.

Les attributs `horaireDepart` et `horaireArrivee` sont de type `TIME` et s'écrivent selon le format `"hh:mm"`, où `"hh"` représente les heures et `"mm"` les minutes.

1. Quel nom générique donne-t-on aux logiciels qui assurent, entre autres, la persistance des données, l'efficacité de traitement des requêtes et la sécurisation des accès pour les bases de données ?

Les SGBD (Systèmes de Gestion de Base de Données) sont des logiciels qui permettent de réaliser des requêtes permettant d'accéder à des bases de données.

2.

a. On considère les requêtes SQL suivantes :

```
DELETE FROM Train WHERE numT = 1241 ;
DELETE FROM Reservation WHERE numT = 1241 ;
```

Sachant que le train n°1241 a été enregistré dans la table `Train` et que des réservations pour ce train ont été enregistrées dans la table `Reservation`, expliquer pourquoi cette suite d'instructions renvoie une erreur.

numT de la table Reservation est une clé étrangère qui fait référence à numT de la table Train. Cette contrainte de clé étrangère impose d'avoir l'attribut numT = 1241 qui existe dans la table Train. Si on supprime l'enregistrement numT = 1241 dans la table Train en premier, cette contrainte de clé étrangère ne sera plus vérifiée, d'où une l'apparition d'une erreur.
Pour supprimer l'enregistrement numT = 1241 il s'agit dans un premier temps d'exécuter la requête DELETE FROM Reservation WHERE numT = 1241 , puis d'exécuter ensuite la requête DELETE FROM Train WHERE numT = 1241 .

b. Citer un cas pour lequel l'insertion d'un enregistrement dans la table `Reservation` n'est pas possible.

Pour poursuivre le propos précédent, on peut dire que l'enregistrement dans la table Reservation ne sera pas possible si l'attribut numT n'existe pas dans la table Train.

3. Écrire des requêtes SQL correspondant à chacune des instructions suivantes :

a. Donner tous les numéros des trains dont la destination est « Lyon ».

SELECT numT FROM Train WHERE destination = 'Lyon'

b. Ajouter une réservation n°1307 de 33 € pour M. Alan Turing dans le train n°654.

INSERT INTO Reservation VALUES (1307 , 'Turing' , 'Alan' , 33 , 654)

c. Suite à un changement, l'horaire d'arrivée du train n°7869 est programmé à 08 h 11. Mettre à jour la base de données en conséquence.

UPDATE Train SET horaireArrivee = '08 :11' WHERE numT = 7869

4. Que permet de déterminer la requête suivante ?

```
SELECT COUNT(*) FROM Reservation
WHERE nomClient = "Hopper" AND prenomClient = "Grace";
```

Cette requête donne le nombre de réservations au nom de Hopper Grace

5. Écrire la requête qui renvoie les destinations et les prix des réservations effectuées par Grace Hopper.

Cette requête est :

```
SELECT t.destination , r.prix FROM Train AS t
JOIN Reservation AS r ON t.numT = t.numT
WHERE r.prenomClient = 'Grace' AND r.nomClient = 'Hopper'
```

Exercice 2. : Un fournisseur internet a besoin de connaitre la localisation de ses clients. Il établit une base de données nommée « *internet.db* ». Le schéma relationnel de cette base est le suivant :

villes (id_town INTEGER , nom TEXT , population INTEGER)

utilisateurs (id_user INTEGER , nom TEXT, prenom TEXT, genre TEXT, age INTEGER, # id_town)

La table *villes* contient les enregistrements suivants :

id_town	nom	population
Filtre	Filtre	Filtre
1	LYON	513000
2	PARIS	2161000
3	MARSEILLE	861000
4	MACON	33000

1. Donner la requête SQL qui permet de créer la table *utilisateurs*

```
CREATE TABLE utilisateurs  (
id_user INTEGER PRIMARY KEY ,
nom TEXT,
prenom TEXT,
genre TEXT,
age INTEGER,
id_towns INTEGER,
FOREIGN KEY(id_towns) REFERENCES villes(id_town)) ;
```

2. Donner la requête SQL qui permet d'insérer dans la table *utilisateurs* Mr DUJARDIN Jean qui habite à Paris. Son âge n'est pas donné, son *id_user* sera égal à 7.

```
INSERT INTO utilisateurs(id_user,nom,prenom,genre,id_towns) VALUES
(7,DUJARDIN,'Jean','homme',2)
```

3. Donner la requête SQL qui permet de compléter l'âge de Mr DUJARDIN Jean dans la table *utilisateurs* : 50 ans.

```
UPDATE utilisateurs SET age = 50 WHERE nom = 'DUJARDIN';
```

4. Donner la requête SQL qui permet de déterminer l'âge moyen des personnes enregistrées dans la table utilisateurs en les regroupant par genre. La colonne donnant l'âge sera notée *ageMoyen*.

	genre	ageMoyen
1	femme	45.0
2	homme	45.75

```
SELECT genre, AVG(age) AS ageMoyen FROM utilisateurs GROUP BY genre;
```

5. Donner la requête SQL qui permet de déterminer les nom, prénom des personnes enregistrées dans la table *utilisateurs* et qui vivent à MACON.

```
SELECT u.nom,u.prenom FROM utilisateurs AS u
JOIN villes AS v ON u.id_towns = v.id_town
WHERE v.nom = 'MACON'
```

6. Donner le résultat dans le shell, de l'exécution du code python ci-dessous :

```
import sqlite3
connexion = sqlite3.connect("internet.db")
curseur = connexion.cursor()

curseur.execute("""
                    SELECT * FROM villes
                """)
resultat = curseur.fetchall()

print(resultat)

connexion.close()
```

L'exécution de ce code donnera :

```
>>> (executing file "ds.py")
[(1, 'LYON', 513000), (2, 'PARIS', 2161000), (3, 'MARSEILLE', 861000), (4, 'MACON', 33000)]
```