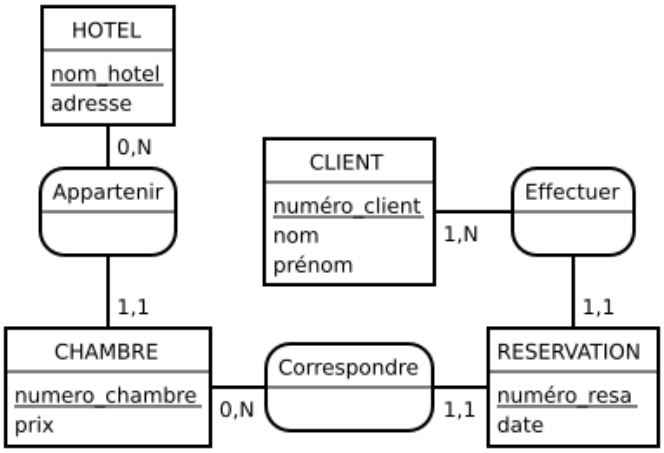
Bases de données

1. Modèle relationnel

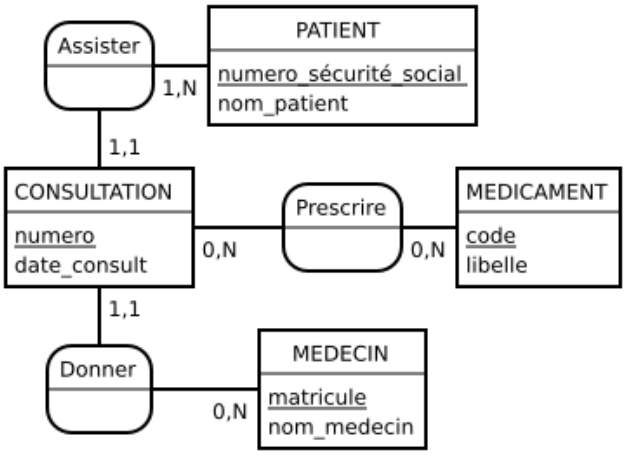
Exercice 1 (Hors programme)
On souhaite modéliser les réservations dans une compagnie d'hôtels. On considère le modèle entité-association suivant :



A l'aide de ce modèle répondre aux questions suivantes :

1. Peut-on avoir des clients homonymes ?
2. Un client peut-il réserver plusieurs chambres à une date donnée ?
3. Est-il possible de réserver une chambre sur plusieurs jours ?
4. Peut-on savoir si une chambre est libre à une date donnée ?
5. Peut-on réserver plusieurs fois une chambre à une date donnée ?

Exercice 2 (Hors programme)
On donne ci-dessous le modèle entité-association représentant des visites dans un centre médical.



En utilisant ce modèle répondre aux questions suivantes :

1. Un patient peut-il effectuer plusieurs visites ?
2. Un médecin peut-il recevoir plusieurs patients dans la même consultation ?
3. Peut-on prescrire plusieurs médicaments dans une même consultation ?
4. Deux médecins différents peuvent-ils prescrire le même médicament ?

Exercice 3

1. Donner le modèle relationnel de la base de données « compagnie d'hôtels » décrite par le modèle entité-association de l'exercice 1.
2. Donner le modèle relationnel de la base de données « centre médical » décrite par le modèle entité-association de l'exercice 2.

Exercice 4

On souhaite modéliser un annuaire téléphonique simple dans lequel chaque personne est associée à son numéro de téléphone. Donner un modèle relationnel pour cet annuaire.

Exercice 5

On souhaite modéliser un bulletin scolaire. La base de données contient trois tables (relations) :

- Une table Eleve. Chaque élève possède en particulier un numéro d'étudiant unique ;
- Une table Matiere ;
- Une table Note.

Sachant que l'on souhaite au plus une note par élève et par matière, donner pour chaque table son schéma.

Exercice 6

Pour chaque département on veut pouvoir obtenir son nom, son code, son chef-lieu et obtenir la liste de tous les départements voisins.
Donner un modèle relationnel permettant de stocker ces informations.

Exercice 7

Proposer un modèle relationnel pour un réseau de bus. Celui-ci doit être suffisamment riche pour permettre de générer, pour chaque arrêt de bus du réseau, une fiche horaire avec tous les horaires de passage de toutes les lignes de bus qui desservent l'arrêt.

Exercice 8

On donne ci-dessous les occurrences de la relation *Consultation* issue du modèle relationnel construit dans l'exercice 3 question 2.

numero	matricule	numero_sécurité_sociale	date_consult
1	123		21/11/2018
2	123	18208692682812	
2	526	Aspirine	13/03/2019

Citer les anomalies constatées.

Exercice 9

On considère la base de données « bulletin » de l'exercice 5.
Dire si chacun des cas suivant donne un ensemble de relations valides pour le schéma de la base de données bulletin de notes.

1.
   - `Eleve = {('Titi','Toto','AB56789')}`
   - `Matiere = {('NSI',0),('Sport',1)}`
   - `Note = {('AB56789',1,17)}`

2.
   - `Eleve = {('Titi','Toto','AB56789')}`
   - `Matiere = {('NSI',0)}`
   - `Note = {('AB56789',1,17)}`
3.
   - `Eleve = {('Titi','Toto','AB56789')}`
   - `Matiere = {('NSI',0)}`
   - `Note = {('AB56789',0,17),('AB56789',0,18)}`
4.
   - `Eleve = {('Titi','Toto','AB56789')}`
   - `Matiere = {('NSI',0),('Sport',1)}`
   - `Note = {('AB56789',1,17),('AB56789',0,17)}`

Exercice 10

Un institut a constitué un tableau contenant des données statistiques sur une épidémie qui s'est répandue dans tous les pays. Ce tableau est constitué de quatre colonnes représentant le nom d'un pays, le numéro d'un jour (de 1 à 365), le nombre de cas confirmés et le nombre de décès. Voici quatre lignes extraites du tableau :

pays	**jour**	**cas**	**décès**
France	83	1195	186
Allemagne	87	966	53
Suisse	95	228	17
France	108	2866	441

Expliquer quelle peut être une clé primaire.

2. Langage SQL

Les exercices 1 à 5 font référence à la base de données `CDI` étudiée en classe. Les mots en `police fixe` donnent une indication sur les tables et attributs à utiliser dans la requête. Attention toutefois aux accents, qui ont été éliminés dans les noms des tables et de leurs attributs. Penser à tester les requêtes à l'aide de sqlite3.

Exercice 1 (Requêtes simples)
Donner le code SQL de chacune des requêtes suivantes.
On utilisera entre autres les instructions `SELECT FROM`, `DISTINCT`, `ORDER BY` et `LIKE`.

1. Afficher tous les `noms` des `auteurs`.
2. Afficher le `titre` de tous les `livres`.
3. Afficher les `noms` des `classes` du lycée sans doublon.
4. Afficher les `titres` des `livres` et les `années` d'édition classé selon l'`année`.
5. Quelles sont les `livres` dont le `titre` contient le mot 'Astérix' ?

Exercice 2 (Expressions et fonctions)
Donner le code SQL de chacune des requêtes suivantes.

1. Afficher Les `noms` et prénoms des `élèves` de la `classe` '1-G1'.
2. Afficher les `titres` des `livres` publiés après 2018.
3. Afficher les `isbn` des `livres` dont la date retour est déjà passée.
4. Combien d'`auteurs` sont présents dans la base de données ?
5. Quelle est l'`année` d'édition du ou des `livre(s)` le(s) plus ancien(s) ?

Exercice 3 (Requêtes imbriquées)
Donner le code SQL de chacune des requêtes suivantes.

1. Afficher les `titres` des `livres` empruntés.
2. Afficher, sans doublon, le `nom` et prénom des `élèves` qui ont emprunté au moins un `livre`.
3. Afficher, sans doublon, le `nom` et prénom des `élèves` qui ont emprunté au moins un `livre` avec une date retour dépassée.
4. Qui est l'auteur du `livre` '1984' ?
5. Quelle est le `titre` du ou des `livre`(s) le(s) plus ancien(s) ?

Exercice 4 (Jointure)
Donner le code SQL de chacune des requêtes suivantes en utilisant la clause `JOIN`.

1. Afficher les `titres` des `livres` empruntés.
2. Afficher, sans doublon, le `nom` et prénom des `élèves` qui ont emprunté au moins un `livre`.
3. Qui est l'auteur du `livre` '1984' ?
4. Quels sont les `éditeurs` à avoir édité un `livre` contenant 'Astérix' dans le `titre` ?
5. Afficher les `titres` des `livres` écrit par un auteur dont le nom contient 'Boulle'.
6. Combien de `livres` ont été écrit par un auteur dont le nom contient 'Asimov' ?
7. Afficher les `noms` des `éditeurs` ayant édité un `livre` écrit par un auteur dont le nom contient 'Barjavel'.

Exercice 5
Formuler en français les requêtes SQL suivante.

1. ```
   SELECT * FROM Livre WHERE titre LIKE '%Robot%';
   ```
2. ```
   SELECT nom, prenom FROM Eleve WHERE classe = 'T-G2';
   ```
3. ```
   SELECT el.nom, el.prenom FROM Eleve AS el
                            JOIN Emprunt AS em ON el.num_etu = em.num_etu
                            WHERE date_ret < '2022-12-20';
   ```

   Réécrire la requête en utilisant la clause `USING`.
4. ```
   SELECT l.titre FROM Livre AS l
                  WHERE l.isbn IN (SELECT isbn FROM Livre WHERE annee > 1990);
   ```

   Réécrire la requête en utilisant une seule clause `SELECT`.

Pour les exercices suivants, on considère les tables décrites ci-dessous :

```
CREATE TABLE x (a INT PRIMARY KEY, b INT, CHECK (b >= 0));
CREATE TABLE y (c INT PRIMARY KEY, d INT, CHECK (d <= 30));
CREATE TABLE z (a INT REFERENCES X(a), c INT REFERENCES Y(c),
                e INT, PRIMARY KEY(a,c));
```

x :

a	b
1	1
2	2
3	2
4	2
5	1
6	9
7	1

y :

c	d
9	9
10	10
11	9
12	20
13	30
14	9
15	1
16	10
17	10

z :

a	c	e
1	11	30
2	14	9
5	15	1
7	17	3
1	10	50
2	9	8
2	15	15
3	17	19
4	16	12
5	10	20
2	11	30
7	14	9
7	9	12

Exercice 6

Pour chacune des requêtes suivantes, calculer son résultat à la main.

1. ```
   SELECT * FROM x WHERE b > 3;
   ```
2. ```
   SELECT DISTINCT e FROM z
                     WHERE e > 10 AND e < 50;
   ```
3. ```
   SELECT * FROM y WHERE c % 2 = 0 ORDER BY d ASC;
   ```
4. ```
   SELECT DISTINCT x.b,y.d FROM x
                           JOIN z ON z.a = x.a
                           JOIN y ON y.c = z.c;
   ```

Exercice 7

Pour chacune des modifications suivantes, indiquer si elle réussit ou si elle échoue. Si elle réussit indiquer comment la table est modifiée. Si elle échoue, expliquer pourquoi.

1. ```
   UPDATE x SET b = b + a;
   ```
2. ```
   UPDATE x SET b = b - 2;
   ```
3. ```
   INSERT INTO z VALUES (1, 17, 1);
   ```
   ```

4. `INSERT INTO z VALUES (1, 18, 1);`
5. `INSERT INTO z VALUES (1, 10, 1);`
6. `DELETE FROM y WHERE c >= 12 AND c <= 13;`
7. `DELETE FROM y WHERE c >= 12 AND c <= 14;`
8. `INSERT INTO y VALUES (40, 20);`
9. `INSERT INTO y VALUES (20, 40);`
10. `DELETE FROM z WHERE a % 2 = 0 OR c % 2 = 0 OR e % 2 = 0;`

Exercice 8

Pour chacune des séquences d'ordres SQL suivantes, dire quelle instruction provoque une erreur. On suppose que la base de donnée ne contient aucune table au début de chaque séquence.

1.
```
DROP TABLE client;
CREATE TABLE client (cid INT PRIMARY KEY,
                     nom VARCHAR(100),
                     prenom VARCHAR(100),
                     points_fidelite INT NOT NULL,
                     CHECK (points_fidelite >= 0));
```

2.
```
CREATE TABLE client (cid INT PRIMARY KEY,
                     nom VARCHAR(100),
                     prenom VARCHAR(100),
                     points_fidelite INT NOT NULL,
                     CHECK (points_fidelite >= 0));

CREATE TABLE commande (cid INT REFERENCES client(cid),
                       pid INT REFERENCES produit(pid),
                       date DATE NOT NULL,
                       PRIMARY KEY(cid, pid));

CREATE TABLE produit (pid INT PRIMARY KEY,
                      nom VARCHAR(100),
                      prix DECIMAL(10,2));
```

3.
```
CREATE TABLE client (cid INT PRIMARY KEY,
                     nom VARCHAR(100),
                     prenom VARCHAR(100),
                     points_fidelite INT NOT NULL,
                     CHECK (points_fidelite >= 0));

CREATE TABLE produit (pid INT PRIMARY KEY,
                      nom VARCHAR(100),
                      prix DECIMAL(10,2));

CREATE TABLE commande (cid INT REFERENCES client(cid),
                       nomp VARCHAR(100) REFERENCES produit(nom),
                       date DATE NOT NULL,
                       PRIMARY KEY(cid, nomp));
```

4.
```
CREATE TABLE client (cid INT PRIMARY KEY,
                     nom VARCHAR(100),
                     prenom VARCHAR(100),
```

```
                        points_fidelite INT NOT NULL,
                        CHECK (points_fidelite >= 0));

CREATE TABLE produit (pid INT PRIMARY KEY,
                      nom VARCHAR(100),
                      prix DECIMAL(10,2));

CREATE TABLE commande (cid INT REFERENCES client(cid),
                       pid INT REFERENCES produit(pid),
                       date DATE NOT NULL,
                       PRIMARY KEY(cid, pid));

INSERT INTO commande VALUES (0, 0, '2020-03-02');
```

Exercice 9
On considère les deux tables suivantes qui stockent des résultats de parties entre joueurs :

```
CREATE TABLE joueur (jid INT PRIMARY KEY,
                     nom VARCHAR(100) NOT NULL);

CREATE TABLE partie (pid INT PRIMARY KEY,
                     j1 INT REFERENCES joueur(jid),
                     j2 INT REFERENCES joueur(jid),
                     score1 INT NOT NULL,
                     score2 INT NOT NULL,
                     CHECK (j1<>j2));
```

Lister et expliquer toutes les contraintes d'intégrité.

Exercice 10
Modifier les ordres de création de l'exercice précédent pour prendre en compte les modifications suivantes :

- La table partie contient en plus une colonne `jour` non nulle, indiquant la date à laquelle la partie à eu lieu.
- Les scores ne peuvent pas être négatifs.
- Deux joueurs ne peuvent pas jouer ensemble plusieurs fois le même jour.

Exercice 11 (Pour les élèves ayant une certaine motivation)
Pour les exercices 3, 4, 5, 6 et 7 de la section 1, donner les ordres SQL permettant de créer les tables correspondant au modèle relationnel avec un maximum de contraintes.

Exercice 12 (Pour les élèves ayant une très forte motivation)
Écrire un programme Python qui lit un fichier CSV `infos.csv` au format suivant :

- les champs sont séparés par des ';' ;
- le fichier contient 4 colonnes `nom`, `prenom`, `annee_naissance` et `taille`.

Le programme doit écrire sur sa sortie standard un script SQL (i.e un ensemble d'ordres) qui :

1. crée une table permettant de stocker ces informations ainsi qu'un entier unique servant de clé primaire ;
2. remplit la table avec les données du fichier CSV.

EXERCICE 3 (4 points)

Cet exercice traite du thème « base de données », et principalement du modèle relationnel et du langage SQL.

L'énoncé de cet exercice peut utiliser les mots du langage SQL suivants :
`CREATE TABLE, SELECT, FROM, WHERE, JOIN ON, INSERT INTO, VALUES, UPDATE, SET, DELETE, COUNT, DISTINCT, AND, OR, AS, ORDER BY, ASC, DESC`

Un site web recueille des données de navigation dans une base de données afin d'étudier les profils de ses visiteurs.
Chaque requête d'interrogation d'une page de ce site est enregistrée dans une première table dénommée **Visites** sous la forme d'un 5-uplet : `(identifiant, adresse IP, date et heure de visite, nom de la page, navigateur)`.

Le chargement de la page index.html par 192.168.1.91 le 12 juillet 1998 à 22h48 aura par exemple été enregistré de la façon suivante :
`(1534, "192.168.1.91", "1998-07-12 22:48:00", "index.html", "Internet explorer 4.1")`.

La commande SQL ayant permis de créer cette table est la suivante:

```
CREATE TABLE Visites (
    identifiant INTEGER NOT NULL UNIQUE,
    ip VARCHAR(15),
    dateheure DATETIME,
    nompage TEXT,
    navigateur TEXT
);
```

1. a. Donner une commande d'interrogation en langage SQL permettant d'obtenir l'ensemble des 2-uplets (adresse IP, nom de la page) de cette table.

   b. Donner une commande en langage SQL permettant d'obtenir l'ensemble des adresses IP ayant interrogé le site, sans doublon.

   c. Donner une commande en langage SQL permettant d'obtenir la liste des noms des pages visitées par l'adresse IP 192.168.1.91

Ce site web met en place, sur chacune de ses pages, un programme en javascript qui envoie au serveur, à intervalle régulier de 15 secondes, le temps en secondes de présence sur la page. Ces envois contiennent tous la valeur de `identifiant` correspondant au chargement initial de la page.
Par exemple, si le visiteur du 12 juillet 1998 est resté 65 secondes sur la page, celle-ci a envoyé au serveur les 4 doublets (1534, 15), (1534, 30), (1534, 45) et (1534, 60).

Ces données sont enregistrées dans une table nommée `Pings` créée avec la commande ci-dessous :

```
CREATE TABLE Pings (
    identifiant INTEGER,
    duree INTEGER
);
```

En plus de l'inscription d'une ligne dans la table `Visites`, chaque chargement d'une nouvelle page provoque l'insertion d'une ligne dans la table `Pings` comprenant l'identifiant de ce chargement et une durée de 0.

Les attributs `identifiant` des tables `Visites` et `Pings` partagent les mêmes valeurs.

2. a. De quelle table l'attribut `identifiant` est-il la clé primaire ?

   b. De quelle table l'attribut `identifiant` est-il une clé étrangère ?

   c. Par conséquent, quelles vérifications sont automatiquement effectuées par le système de gestion de base de données ?

3. Le serveur reçoit le doublet `(identifiant, duree)` suivant : (1534, 105). Écrire la commande SQL d'insertion qui permet d'ajouter cet enregistrement à la table `Pings`.

On envisage ensuite d'optimiser la table en se contentant d'une seule ligne par identifiant dans la table `Pings` : les valeurs de l'attribut `duree` devraient alors être mises à jour à chaque réception d'un nouveau doublet `(identifiant, duree)`.

4. a. Écrire la requête de mise à jour permettant de fixer à 120 la valeur de l'attribut `duree` associée à l'identifiant 1534 dans la table `Pings`.

   b. Expliquer pourquoi on ne peut pas être certain que les données envoyées par une page web, depuis le navigateur d'un client, via plusieurs requêtes formulées en javascript, arrivent au serveur dans l'ordre dans lequel elles ont été émises.

   c. En déduire qu'il est préférable d'utiliser une requête d'insertion plutôt qu'une requête de mise à jour pour ajouter des données à la table `Pings`.

5. Écrire une requête SQL utilisant le mot-clef `JOIN` et une clause `WHERE`, permettant de trouver les noms de toutes les pages qui ont été consultées plus d'une minute par au moins un utilisateur.

EXERCICE 2 (4 points)

Cet exercice porte sur les bases de données.

On pourra utiliser les mots clés SQL suivants : **SELECT**, **FROM**, **WHERE**, **JOIN**, **ON**, **INSERT**, **INTO**, **VALUES**, **UPDATE**, **SET**, **AND**.

Nous allons étudier une base de données traitant du cinéma dont voici le schéma relationnel qui comporte 3 relations :

- la relation **individu** (id_ind, nom, prenom, naissance)
- la relation **realisation** (id_rea, titre, annee, type)
- la relation **emploi** (id_emp, description, #id_ind, #id_rea)

Les clés primaires sont soulignées et les clés étrangères sont précédées d'un #.
Ainsi **emploi**.id_ind est une clé étrangère faisant référence à **individu**.id_ind.
Voici un extrait des tables **individu** et **realisation** :

extrait de **individu**				extrait de **realisation**			
id_ind	nom	prenom	naissance	id_rea	titre	annee	type
105	'Hulka'	'Daniel'	'01-06-1968'	105	'Casino Imperial'	2006	'action'
403	'Travis'	'Daniel'	'10-03-1968'	325	'Ciel tombant'	2012	'action'
688	'Crog'	'Daniel'	'07-07-1968'	655	'Fantôme'	2015	'action'
695	'Pollock'	'Daniel'	'24-08-1968'	950	'Mourir pour attendre'	2021	'action'

1. On s'intéresse ici à la récupération de données dans une relation.

a. Écrire ce que renvoie la requête ci-dessous :

```
SELECT nom, prenom, naissance
FROM individu
WHERE nom = 'Crog';
```

b. Fournir une requête SQL permettant de récupérer le titre et la clé primaire de chaque film dont la date de sortie est strictement supérieure à 2020.

2. Cette question traite de la modification de relations.

a. Dire s'il faut utiliser la requête 1 ou la requête 2 proposées ci-dessous pour modifier la date de naissance de Daniel Crog. Justifier votre réponse en expliquant pourquoi la requête refusée ne pourra pas fonctionner.

Requête 1

```
UPDATE individu
SET naissance = '02-03-1968'
WHERE id_ind = 688 AND nom = 'Crog' AND prenom = 'Daniel';
```

Requête 2

```
INSERT INTO individu
VALUES (688, 'Crog', 'Daniel', '02-03-1968');
```

b. Expliquer si la relation **`individu`** peut accepter (ou pas) deux individus portant le même nom, le même prénom et la même date de naissance.

3. Cette question porte sur la notion de clés étrangères.

a. Recopier sur votre copie les demandes ci-dessous, dans leur intégralité, et les compléter correctement pour qu'elles ajoutent dans la relation **`emploi`** les rôles de Daniel Crog en tant que James Bond dans le film nommé 'Casino Impérial' puis dans le film 'Ciel tombant'.

```
INSERT INTO emploi
VALUES (5400, 'Acteur(James Bond)', ... );

INSERT INTO emploi
VALUES (5401, 'Acteur(James Bond)', ... );
```

b. On désire rajouter un nouvel emploi de Daniel Crog en tant que James Bond dans le film 'Docteur Yes'.
Expliquer si l'on doit d'abord créer l'enregistrement du film dans la relation **`realisation`** ou si l'on doit d'abord créer le rôle dans la relation emploi.

4. Cette question traite des jointures.

a. Recopier sur votre copie la requête SQL ci-dessous, dans son intégralité, et la compléter de façon à ce qu'elle renvoie le nom de l'acteur, le titre du film et l'année de sortie du film, à partir de tous les enregistrements de la relation emploi pour lesquels la description de l'emploi est `'Acteur(James Bond)'`.

```
SELECT ...
FROM emploi
JOIN individu ON ...
JOIN realisation ON ...
WHERE emploi.description = 'Acteur(James Bond)';
```

b. Fournir une requête SQL permettant de trouver toutes les descriptions des emplois de Denis Johnson (Denis est son prénom et Johnson est son nom). On veillera à n'afficher que la description des emplois et non les films associés à ces emplois.

EXERCICE 4 (4 points) :

Cet exercice porte sur les bases de données

Un rappel sur la syntaxe de quelques fonctions SQL est donné en annexe 1 en fin de sujet.

Dans le cadre d'une étude sur le réchauffement climatique, un centre météorologique rassemble des données. On considère que la base de données contient deux relations (tables). La relation `Centres` qui contient l'identifiant des centres météorologiques, la ville, la latitude, la longitude et l'altitude du centre. La relation `Mesures` qui contient l'identifiant de la mesure, l'identifiant du centre, la date de la mesure, la température, la pression et la pluviométrie mesurées.

Le schéma relationnel de la relation `Centres` est le suivant :

```
Centres(id_centre: INT, nom_ville: VARCHAR, latitude: FLOAT,
longitude: FLOAT, altitude: FLOAT)
```

Le schéma relationnel de la relation `Mesures` est le suivant :

```
Mesures(id_mesure: INT, id_centre: INT, date: DATE,
temperature: FLOAT, pression: INT, pluviometrie: FLOAT).
```

Relation `Centres`

id_centre	nom_ville	latitude	longitude	altitude
213	Amiens	49.894	2.293	60
138	Grenoble	45.185	5.723	550
263	Brest	48.388	-4.49	52
185	Tignes	45.469	6.909	2594
459	Nice	43.706	7.262	260
126	Le Puy-en-Velay	45.042	3.888	744
317	Gérardmer	48.073	6.879	855

Relation `Mesures`

id_mesure	id_centre	date	temperature	pression	pluviometrie
1566	138	2021-10-29	8.0	1015	3
1568	213	2021-10-29	15.1	1011	0
2174	126	2021-10-30	18.2	1023	0
2200	185	2021-10-30	5.6	989	20
2232	459	2021-10-31	25.0	1035	0
2514	213	2021-10-31	17.4	1020	0
2563	126	2021-11-01	10.1	1005	15
2592	459	2021-11-01	23.3	1028	2
3425	317	2021-11-02	9.0	1012	13
3430	138	2021-11-02	7.5	996	16
3611	263	2021-11-03	13.9	1005	8
3625	126	2021-11-03	10.8	1008	8

1.

a. Proposer une clé primaire pour la relation `Mesures`. Justifier votre choix.

b. Avec quel attribut peut-on faire une jointure entre la relation `Centres` et la relation `Mesures` ?

2.

a. Qu'affiche la requête suivante ?

```
SELECT * FROM Centres WHERE altitude>500;
```

b. On souhaite récupérer le nom de la ville des centres météorologiques situés à une altitude comprise entre 700m et 1200m. Ecrire la requête SQL correspondante.

c. On souhaite récupérer la liste des longitudes et des noms des villes des centres météorologiques dont la longitude est supérieure à 5. La liste devra être triée par ordre alphabétique des noms de ville. Ecrire la requête SQL correspondante.

3.

a. Qu'affiche la requête suivante ?

```
SELECT * FROM Mesures WHERE date="2021-10-30";
```

b. Écrire une requête SQL permettant d'ajouter une mesure prise le 8 novembre 2021 dans le centre numéro 138, où la température était de 11°C, la pression de 1013 hPa et la pluviométrie de 0mm. La donnée dont l'attribut est `id_mesure` aura pour valeur 3650.

4.

a. Expliquer ce que renvoie la requête SQL suivante ?

```
SELECT * FROM Centres WHERE latitude = (SELECT
  MIN(latitude) FROM Centres);
```

b. Écrire une requête SQL donnant la liste des villes dans lesquelles on a enregistré une température inférieure à 10°C en octobre 2021. On utilisera le mot clé `DISTINCT` afin d'éviter d'avoir des doublons. On rappelle que l'on peut utiliser les opérateurs de comparaison avec les dates.

ANNEXE 1 – LANGAGE SQL

- **Types de données**

CHAR(t) VARCHAR(t) TEXT	Texte fixe de t caractères. Texte de t caractères variables. Texte de 65 535 caractères max.
INT	*Nombre entier de* -2^{31} à $2^{31}-1$ (signé) ou de 0 à $2^{32}-1$ (non signé)
FLOAT	Réel à virgule flottante
DATE	Date format AAAA-MM-JJ

- **Quelques exemples de syntaxe SQL :**
  - Insérer des enregistrements :

```
INSERT INTO Table (attribut1, attribut2) VALUES(valeur1 , valeur2)
```

  - Modifier des enregistrements :

```
UPDATE Table SET attribut1=valeur1, attribut2=valeur2 WHERE Selecteur
```

  - Supprimer des enregistrements :

```
DELETE FROM Table WHERE Selecteur
```

  - Sélectionner des enregistrements :

```
SELECT attributs FROM Table WHERE Selecteur
```

  - Sélectionner des enregistrements dans un ordre ascendant :

```
SELECT attributs FROM Table WHERE Selecteur ORDER BY attribut ASC
```

  - Sélectionner des enregistrements sans doublon :

```
SELECT DISTINCT attributs FROM Table WHERE Selecteur
```

  - Effectuer une jointure :

```
SELECT attributs FROM TableA JOIN TableB ON TableA.cle1=TableB.cle2 WHERE
Selecteur
```

Exercice 2

Thème abordé : bases de données

Un restaurant décide de créer son site de réservation en ligne pour son unique service du midi. Voici le schéma relationnel de la base de données imaginée par le concepteur du site. Elle est composée de 4 relations (tables) :

```
plat(id_plat, nom_plat, type_plat, prix_plat)
table_salle(num_table, nb_couvert_table, type_table)
client(num_client, nom_client, prenom_client, date_naiss_client,
mel_client, tel_client)
reservation(num_reserv, nb_pers_reserv, date_reserv, num_table, num_client)
```

Les chaînes de caractères dans cette base n'excèdent pas 100 caractères.
Le `num_client` et le `num_reservation` sont incrémentés automatiquement.
Voici un exemple possible d'enregistrement de la relation `plat` :

id_plat	nom_plat	type_plat	prix_plat
14	"brownie"	"Dessert"	6.35

1. Etude du schéma relationnel

a. Proposer pour chaque attribut de la relation `plat` son domaine (type), on pourra s'aider de l'annexe 1 : mémento SQL

b. Pour chacune des 4 relations, écrire les noms des clés primaires pouvant être utilisées.

c. Indiquer la ou les clés étrangères de la relation `reservation`. Préciser l'utilité d'une clé étrangère.

En vous aidant de l'ANNEXE 1 : memento SQL

2. Recherche et modification d'informations dans la base de données

a. Ecrire une requête SQL permettant d'afficher le nom des plats, leur type et leur prix.

b. Ecrire une requête SQL permettant d'afficher les noms de tous les plats proposés par le restaurant qui sont des desserts.

c. Les coordonnées d'un client ont été enregistrées :

num_client	nom_client	prenom_client	date_naiss_client	mel_client	tel_client
42	Martin	Fiona	1998-03-17	fmartin@laposte.net	"0605040302"

Ecrire la requête SQL permettant de modifier le numéro de téléphone de ce client en "0602030405".

d. Ecrire la requête SQL permettant d'afficher le nom de tous les clients ayant déjà mangé à la table n°13.

Annexe 1 (exercice 2)

(à ne pas rendre avec la copie)

- **Types de données**

CHAR(t) VARCHAR(t) TEXT	Texte fixe de t caractères. Texte de t caractères variables. Texte de 65 535 caractères max.
INT	*Nombre entier de* -2^{31} à $2^{31}-1$ (signé) ou de 0 à $2^{32}-1$ (non signé)
FLOAT	Réel à virgule flottante
DATE DATETIME	Date format AAAA-MM-JJ Date et heure format AAAA-MM-JJHH:MI:SS

- **Quelques exemples de syntaxe SQL :**
  - Insérer des enregistrements :

```
INSERT INTO Table (attribut1, attribut2) VALUES(valeur1 , valeur2)
```

  - Modifier des enregistrements :

```
UPDATE Table SET attribut1=valeur1, attribut2=valeur2 WHERE Selecteur
```

  - Supprimer des enregistrements :

```
DELETE FROM Table WHERE Selecteur
```

  - Sélectionner des enregistrements :

```
SELECT attributs FROM Table WHERE Selecteur
```

  - Effectuer une jointure :

```
SELECT attributs FROM TableA JOIN TableB ON TableA.cle1=TableB.cle2 WHERE Selecteur
```