Exercice 1. 1. Pour les exercices 3, 4 et 6 de la feuille d'exercices *modèle relationnel*, donner les ordres SQL permettant de créer les tables correspondantes à la modélisation avec un maximum de contraintes.

2. Même question pour les exercices 1 et 2.

Exercice 2.
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) RFERENCES produit(nom),
                       date DATE NOT NULL,
                       PRIMARY KEY(cid, pid));
```

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),
                       nomp VARCHAR(100) RFERENCES produit(nom),
                       date DATE NOT NULL,
                       PRIMARY KEY(cid, pid));

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

Exercice 3.
On considère les deux tables suivantes :

```
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));
```

Ces tables stockent des résultats de parties entre joueurs. Lister toutes les contraintes d'intégrité et pour chacune donner des ordres SQL violant ces contraintes.

Exercice 4.
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 être négatifs.
- Deux joueurs ne peuvent pas jouer plusieurs fois le même jour.

Exercice 5.
Ecrire 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 auinsi qu'un entier unique servant de clé primaire et (2) remplit la table avec les données du fichier CSV.