Exercice 1. — Regrouper ensemble les termes synonymes : colonne, entité, domaine, attribut, ligne, schéma, base de données, type, *column*, *row*.

Exercice 2. — Reprendre la modélisation d'un annuaire obtenue en correction de l'exercice 1 du TD7.
Donner un ordre SQL permettant de créer la table correspondante, avec un maximum de contrainte d'intégrité.

Exercice 3. — Reprendre la modélisation d'un bulletin scolaire obtenue en correction de l'exercice 2 du TD7.
Donner les ordres SQL permettant de créer les tables correspondantes, avec un maximum de contrainte d'intégrité.
Donner les ordres SQL permettant de supprimer ces tables une fois qu'elles existent.

Exercice 4. — Pour chacune des séquences d'ordres SQL suivantes, dire quelle instruction provoque une erreur.
On suppose que la base de données ne contient aucune table au début de chaque séquence.

1. ```
   DROP TABLE client;

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

2. ```
   CREATE TABLE client (cid INT PRIMARY KEY,
                        nom VARCHAR(100),
                        prénom 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);

   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),
                        prénom 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),
                          nom VARCHAR(100) REFERENCES produit(nom),
                          date DATE NOT NULL);
   ```

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

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

Exercice 5. — On considère les deux tables suivantes :

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

CREATE TABLE partie (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 des joueurs. Lister toutes les contraintes d'intégrité et pour chacune donner des ordres SQL violant ces contraintes.

Exercice 6. — Modifier les ordres de création de table 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 plusieurs fois le même jour.

Exercice 7. — 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`, `prénom`, `annee_naissance`, `taille`, représentant le nom, prénom, l'année de naissance et la taille (en cm) de personnes.

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

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