```
CREATE DATABASE tpnote;
use tpnote;

CREATE TABLE batiment (
   id_batiment INT AUTO_INCREMENT PRIMARY KEY,
   nom_batiment VARCHAR(10) NOT NULL
);


CREATE TABLE appartement (
   id_appartement INT AUTO_INCREMENT PRIMARY KEY,
   num_appart CHAR(3) NOT NULL,
   etage_appart INT NOT NULL,
   id_batiment INT NOT NULL,

   CONSTRAINT uq_appart UNIQUE (id_batiment, num_appart),
   CONSTRAINT fk_appart_bat FOREIGN KEY (id_batiment) REFERENCES
batiment(id_batiment)
);

CREATE TABLE parking (
   id_parking INT AUTO_INCREMENT PRIMARY KEY,
   num_place CHAR(3) NOT NULL,
   id_batiment INT NOT NULL,
   id_appartement INT NULL,

   CONSTRAINT uq_parking UNIQUE (id_batiment, num_place),
   CONSTRAINT fk_park_bat FOREIGN KEY (id_batiment) REFERENCES
batiment(id_batiment),
   CONSTRAINT fk_park_app FOREIGN KEY (id_appartement) REFERENCES
appartement(id_appartement)
);


INSERT INTO batiment (nom_batiment) VALUES ('A');
INSERT INTO batiment (nom_batiment) VALUES ('B');
INSERT INTO batiment (nom_batiment) VALUES ('C');
INSERT INTO batiment (nom_batiment) VALUES ('D');
INSERT INTO batiment (nom_batiment) VALUES ('E');


INSERT INTO appartement (num_appart, etage_appart, id_batiment) VALUES ('101', 1, 1);
INSERT INTO appartement (num_appart, etage_appart, id_batiment) VALUES ('202', 2, 1);
INSERT INTO appartement (num_appart, etage_appart, id_batiment) VALUES ('103', 1, 2);
INSERT INTO appartement (num_appart, etage_appart, id_batiment) VALUES ('304', 3, 3);
INSERT INTO appartement (num_appart, etage_appart, id_batiment) VALUES ('405', 4, 4);
```

INSERT INTO parking (num_place, id_batiment, id_appartement) VALUES ('001', 1, 1);
INSERT INTO parking (num_place, id_batiment, id_appartement) VALUES ('002', 1, 1);

INSERT INTO parking (num_place, id_batiment, id_appartement) VALUES ('003', 1, 2);

INSERT INTO parking (num_place, id_batiment, id_appartement) VALUES ('001', 2, 3);

INSERT INTO parking (num_place, id_batiment, id_appartement) VALUES ('001', 3, 4);

-- Appart 405 (D) a 2 places
INSERT INTO parking (num_place, id_batiment, id_appartement) VALUES ('001', 4, 5);
INSERT INTO parking (num_place, id_batiment, id_appartement) VALUES ('002', 4, 5);

/* Requêtes :
A. Lister les appartements par ordre croissant de leur numéro
B. Lister, pour chaque appartement, les places de parking associés
C. Afficher le nombre de place de parking par appartement (tous les appartements doivent apparaitre)
D. Afficher uniquement les places de parking non achetés (aucun appartement associé)
E. Pour chaque appartement, compter le nombre de places de parking associées
F. Afficher l'appartement avec le plus de place de parking
G. Afficher la place de parking, occupée par le plus d'app */

/*A - Lister les appartements par ordre croissant de leur numéro*/
SELECT num_appart from appartement ORDER BY num_appart; -- asc optionnel

/*B - Lister, pour chaque appartement, les places de parking associés*/
SELECT appartement.id_appartement, appartement.num_appart,
batiment.nom_batiment, parking.num_place from appartement
JOIN batiment ON appartement.id_batiment = batiment.id_batiment
LEFT JOIN parking ON appartement.id_appartement = parking.id_appartement
ORDER BY appartement.num_appart;

/*C Afficher le nombre de place de parking par appartement (tous les appartements doivent apparaitre)*/
SELECT appartement.id_appartement, appartement.num_appart,
batiment.nom_batiment, parking.num_place from appartement
JOIN batiment ON appartement.id_batiment = batiment.id_batiment
LEFT JOIN parking ON appartement.id_appartement = parking.id_appartement
group by appartement.id_appartement, appartement.num_appart, parking.id_parking
order by appartement.num_appart;

/*D - Afficher uniquement les places de parking non achetés (aucun appartement associé)*/
SELECT parking.id_parking, parking.num_place,batiment.nom_batiment FROM parking
JOIN batiment on parking.id_batiment = batiment.id_batiment
where parking.id_appartement is NULL;

/*E- Pour chaque appartement, compter le nombre de places de parking associées*/
SELECT appartement.id_appartement,appartement.num_appart,batiment.nom_batiment,COUNT(parking.id_parking) AS nb_places FROM appartement
JOIN batiment ON appartement.id_batiment = batiment.id_batiment
LEFT JOIN parking ON appartement.id_appartement = parking.id_appartement
GROUP BY appartement.id_appartement, appartement.num_appart, batiment.nom_batiment
ORDER BY batiment.nom_batiment, appartement.num_appart;

/*F - Afficher l'appartement avec le plus de place de parking*/
/*G*/
/*I*/

NO SQL

Contexte : Une bibliothèque souhaitent faire une système pour suivre l'emprunt des livres de celle-ci, par les abonnés Un abonné peut être identifié par son nom et son prénom mais il peut aussi fournir d'autre information non essentielle (date de naissance, ville, …). Un abonné peut emprunter plusieurs livre en même temps, il réalise des emprunts différents d'un seul livres. Un emprunts est réalisé à une date, aucun historique d'emprunts est conservé, on supprime l'emprunts au rendu par l'abonné. Une livre est identifié par un titre, un auteur mais l'état rend parfois difficile la lecture de ses informations

Prérequis :

use bibliotheque;
db.createCollection("abonnes");

- insérer un abonnés Dupont Jean et son emprunt

```
db.abonnes.insertOne({
   nom: "Dupont",
   prenom: "Jean",
   emprunts: [
      {
         titre: "Les Misérables",
         auteur: "Victor Hugo",
      }
   ]
});
```

Requêtes
-1. À l'aide du contexte ci-dessus, créer le schéma de la base de données NoSQL associé
2. Requêtes :

A. Afficher uniquement le nom et prénom des abonnés

```
db.abonnes.find({},
   { _id: 0, nom: 1, prenom: 1 }
);
```

B. Afficher les personnes qui ont emprunté « Les Misérables »

```
db.abonnes.find({},
{"emprunts.titre": "Les Misérables"}
   { _id: 0, nom: 1, prenom: 1 }
);
```

C. Afficher uniquement les emprunts, trié par ordre alphabétique des titres de livre

```
db.abonnes.aggregate([
{$emprunts}.sort{"emprunts.titre":}},
 {_id:0, titre : "$emprunts.titre", auteur :"$emprunts.auteur"}
]);
```

D. Afficher les abonnés qui ont actuellement au moins un emprunt en cours.

```
db.abonnes.find(
   { "emprunts":
   { _id: 0, nom: 1, prenom: 1 }
);
```

E. Afficher les personnes qui ont emprunté le même livres que le 1er livre emprunté par Dupont Jean

I. Récupérer les emprunts de Dupont Jean
II. Récupérer le premier
III. Faire le filtrage