Exercice 1

Écrire à chaque question une requête permettant de connaître :

1) les villes de Bretagne, en ordonnant les résultats par ordre croissant d'habitants ;

```
SELECT Nom, Habitants FROM Villes
WHERE Region = 'Bretagne'
ORDER BY Habitants
```

2) le plus grand code postal des villes de Bretagne ;

```
SELECT MAX(‘Code postal‘)  FROM Villes
WHERE Region='Bretagne'
```

Remarque : parfois la fonction `MAX` ne fonctionne pas, je ne comprends pas pourquoi, quand je retape la même requête la deuxième (ou troisième) fois ça marche.

3) les villes qui ne sont pas une capitale de région ;

```
SELECT * FROM Villes
WHERE Cle NOT IN (SELECT Capitale FROM Regions)
```

On pourrait aussi chercher les codes postaux non divisibles par 1000.

4) les villes de Bretagne plus peuplées que la moyenne des villes de Bretagne ;

```
SELECT * FROM Villes
WHERE Region = 'Bretagne'
AND Habitants > (SELECT AVG(Habitants) FROM Villes
                 WHERE Region='Bretagne')
```

5) les régions ayant le plus de villes (dans la base).

```
SELECT Region FROM Villes
GROUP BY Region
HAVING COUNT(Cle) =  (SELECT MAX(NB_Villes) FROM
                            (SELECT COUNT(Cle) AS NB_Villes FROM Villes
                             GROUP BY Region) AS VV)
```

Exercice 2 Jumelages

1) Écrire une requête permettant de connaître les villes jumelées avec Rennes.

```
SELECT Nom FROM Villes JOIN Jumelages On Cle=Ville1
WHERE Ville2 IN (SELECT Cle FROM Villes WHERE Nom='Rennes')
UNION
SELECT Nom FROM Villes JOIN Jumelages On Cle=Ville2
WHERE Ville1 IN (SELECT Cle FROM Villes WHERE Nom='Rennes')
```

2) Écrire une requête permettant de connaître les villes jumelées à au moins une ville de Bretagne.

```
SELECT Nom FROM Villes JOIN Jumelages On Cle=Ville1
WHERE Ville2 IN (SELECT Cle FROM Villes WHERE Region='Bretagne')
UNION
SELECT Nom FROM Villes JOIN Jumelages On Cle=Ville2
WHERE Ville1 IN (SELECT Cle FROM Villes WHERE Region='Bretagne')
```

3) Écrire une requête permettant de connaître les villes jumelées à toutes les villes de Bretagne.

Une ville v vérifie cette condition si et seulement s'il n'existe pas de ville de Bretagne w qui ne soit pas jumelée avec v.

```
SELECT Nom FROM Villes AS V
WHERE NOT EXISTS (SELECT * FROM Villes AS W
                  WHERE Region = 'Bretagne'
                  AND NOT EXISTS (SELECT * FROM Jumelages
                                  WHERE (Ville1 = W.Cle AND Ville2=V.Cle)
                                  OR (Ville2=W.Cle AND Ville1=V.Cle)))
```

4) Écrire une requête qui permet de vérifier que la condition sur l'unicité des jumelages est bien respectée dans la table.

La condition est vérifiée si et seulement si la requête ci–dessous renvoie 0.

```
SELECT COUNT(*) FROM Jumelages AS J1, Jumelages AS J2
WHERE J1.Ville1=J2.Ville2 AND J1.Ville2=J2.Ville1
```

Exercice 3 Une pizzeria

1) Quel est le prix de la pizza Calzone ?

```
SELECT Prix FROM Pizzas
WHERE Nom = 'Calzone'
```

2) Quel est le prix moyen des pizzas commandées par M. Athanase Johnovitch ? (on suppose qu'un seul client porte ce nom).

```
SELECT AVG(Pizzas.Prix)
FROM Clients
JOIN Commandes ON Clients.Cle = Commandes.Client
JOIN Pizzas ON Commandes.Pizza = Pizzas.Cle
WHERE Clients.Prenom = 'Athanase'
AND Clients.Nom = 'Johnovitch'
```

3) Retrouver les coordonnées d'un client prénommé Alfred qui n'a toujours pas payé sa commande du mois de novembre.

```
SELECT Prenom, Nom, Adresse, Ville
FROM Clients
JOIN Commandes ON Clients.Cle = Client
WHERE Prenom = 'Alfred'
AND MONTH(Date) = 11
AND Paye = 0
```

4) Quelle est la pizza qui a le plus de succès ?

On trie les pizzas par nombre décroissant de commandes et on ne garde que la première ligne grâce à `LIMIT 0,1`.

```
SELECT Pizzas.Nom, COUNT(Commandes.Cle) AS NbCommandes
FROM Pizzas
JOIN Commandes ON Pizzas.Cle = Commandes.Pizza
GROUP BY Pizzas.Nom
ORDER BY NbCommandes DESC
LIMIT 0, 1
```

5) Quelle est la recette du mois d'avril (en ne comptant que les commandes effectivement payées).

```
SELECT SUM(Prix) FROM Commandes
JOIN Pizzas ON Commandes.Pizza = Pizzas.Cle
WHERE MONTH (Commandes.Date) = 4
AND Commandes.Paye = 1
```

6) Combien de pizzas chaque client a–t–il commandé en moyenne ? Comment savoir si les habitants de Vitré commandent plus de pizzas que la moyenne ?

Première solution pour compter le nombre de pizzas que chaque client a commandé en moyenne :

```
SELECT AVG( C.NbCommandes )
FROM (SELECT COUNT(*) AS NbCommandes
      FROM Commandes
      GROUP BY Client) AS C
```

Deuxième solution : on pourrait compter le nombre de commandes et diviser par le nombre de clients...

Pour calculer le nombre moyen de pizzas commandées par les habitants de Vitré :

```
SELECT AVG(C.NbCommandes)
FROM (SELECT COUNT(*) AS NbCommandes
      FROM Commandes
      JOIN Clients ON Client = Clients.Cle
      WHERE Ville = 'Vitre'
      GROUP BY Client) AS C
```

7) Les pizzas Calzone et Regina sont les seules qui contiennent de la viande. Donner la liste des clients potentiellement végétariens.

```
SELECT Prenom, Nom FROM Clients AS C
WHERE NOT EXISTS (SELECT * FROM Commandes
                          JOIN Pizzas ON Pizza = Pizzas.Cle
                  WHERE C.Cle = Client
                  AND (Pizzas.Nom = 'Calzone' OR Pizzas.Nom = 'Regina'))
```

8) Quelles sont les clients qui ont testé au moins une fois chaque pizza ? Formaliser cela sous forme d'une division sans écrire la requête, puis traduire la requête en SQL.

Soit R la relation (abstraite) dont les deux attributs sont : identification du client, pizza commandée par ce client. Soit S la relation donnant toutes les pizzas. Il s'agit de faire la division de R par S.

On traduit l'énoncé avec "il n'existe pas" : il s'agit de trouver les clients pour lesquels il n'existe aucune pizza qu'ils n'ont pas commandée.

```
SELECT Prenom, Clients.Nom FROM Clients
WHERE NOT EXISTS (SELECT Cle FROM Pizzas
                  WHERE NOT EXISTS (SELECT * FROM Commandes
                                    WHERE Client = Clients.Cle
                                    AND Pizza = Pizzas.Cle))
```

9)
```
SELECT SUM (Nombre * Prix)
FROM Commandes
JOIN CompositionCommande ON Commandes.Cle = CompositionCommande.Commande
JOIN Pizzas ON CompositionCommande.Pizza = Pizzas.Cle
WHERE Commandes.Cle = 1830
```

10) Il semble logique de désigner la liste d'attributs `(Commande,Pizza)` comme clé primaire de cette relation car pour une commande c donnée et une pizza p donnée, il y a logiquement au plus un seul uplet dans la relations `CompositionCommande` correspondant à cette commande et à cette pizza ; par contre il peut y avoir plusieurs uplets correspondant à la commande c, et plusieurs uplets correspondant à la pizza p.

11) On désire maintenant tarifer la livraison en fonction de la ville de livraison (qui correspond peu ou prou à la longueur du trajet).

Proposer une nouveau schéma relationnel. Il faudra pouvoir répondre à la question : "quel est la prix de la commande numéro n dont le client habite à Betton" ? (écrire effectivement cette requête dans votre schéma).