❶ MIDI'HOME CAMPING™ – Révisions du premier semestre – SELECT FROM WHERE

R01 : nom et prénom des clients qui habitent dans la ville de Montpellier classés dans l'ordre lexicographique de leur nom.

```
SELECT nomClient, prenomClient
FROM Clients
WHERE villeClient = 'Montpellier'
ORDER BY nomClient [ASC];
```

Rappeler les options du ORDER BY : ASC, DESC

R02 : l'identifiant, le nom et le prénom des clients qui ont réalisé au moins une location qui a couté plus de 1 000 €.

```
SELECT idClient, nomClient, prenomClient
FROM Clients
WHERE idClient IN (SELECT idClient
                   FROM Locations
                   WHERE montantLocation > 1000);
```

Attention, les étudiants doivent bien comprendre que la requête suivante n'affichera pas le même résultat.

```
SELECT c.idClient, nomClient, prenomClient
FROM Clients c
[INNER] JOIN Locations l ON c.idClient = l.idClient
WHERE montantLocation > 1000;
```

Si on veut le même résultat qu'avec la requête imbriquée, il faut un DISTINCT dans le SELECT

```
SELECT DISTINCT c.idClient, nomClient, prenomClient
FROM Clients c
[INNER] JOIN Locations l ON c.idClient = l.idClient
WHERE montantLocation > 1000;
```

```
Ou encore (avec jointure SQL avant 1992)
```

```
SELECT DISTINCT c.idClient, nomClient, prenomClient
FROM Clients c, Locations l
WHERE c.idClient = l.idClient
AND montantLocation > 1000;
```

Attention s'il n'y a pas l'idClient dans le SELECT et que deux clients ont les mêmes nom et prénom (ce qui va forcement arriver) la solution avec le DISTINCT ne fonctionne pas. Il faut passer par une requête imbriquée. Par exemple :

R02B : le nom et le prénom des clients qui ont réalisé au moins une location qui a couté plus de 1 000 €.

```
SELECT idClient, nomClient, prenomClient
FROM Clients
WHERE idClient IN (SELECT idClient
                   FROM Locations
                   WHERE montantLocation > 1000);
```

Ici, les deux requêtes qui suivent n'afficheront pas le même résultat que celle avec une requête imbriquée

```
SELECT nomClient, prenomClient
FROM Clients c
JOIN Locations l ON c.idClient = l.idClient
WHERE montantLocation > 1000;
```

```
SELECT DISTINCT nomClient, prenomClient
FROM Clients c
JOIN Locations l ON c.idClient = l.idClient
WHERE montantLocation > 1000;
```

Si les étudiants ont bien compris ce que fait une jointure au premier semestre, ils devraient savoir quand il faut mettre ou non un DISTINCT (notamment lorsqu'on fait des jointures). Si ce n'est pas le cas, il faudra insister là dessus. Et peut-être leur réexpliquer ce que fait une jointure.

Par exemple, ils doivent comprendre que le DISTINCT n'est pas automatique. Et qu'on ne le met pas au hasard.

R02C : l'identifiant et la date de début des locations qui ont été passées par un client qui se prénome 'Agathe'

Ici, pas besoin de DISTINCT, les deux requêtes suivantes afficheront le même résultat

```
SELECT idLocation, dateDebut
FROM Locations l
WHERE idClient IN (SELECT idClient
                   FROM Clients
                   WHERE prenomClient = 'Agathe');
```

```
SELECT idLocation, dateDebut
FROM Locations l
JOIN Clients c l ON c.idClient = l.idClient
WHERE prenomClient = 'Agathe';
```

R03 : l'identifiant et le nom des clients Montpelliérains qui ont loué un bungalow qui propose le service Climatisation.

```
SELECT DISTINCT c.idClient, nomClient
FROM Clients c
     JOIN Locations l ON c.idClient = l.idClient
     JOIN Proposer p ON l.idBungalow = p.idBungalow
     JOIN Services s ON p.idService = s.idService
WHERE nomService = 'Climatisation'
AND villeClient = 'Montpellier';
```

Ou

```
SELECT idClient, nomClient
FROM Clients
WHERE villeClient = 'Montpellier'
AND idClient IN (SELECT idClient
                 FROM Locations l
                      JOIN Proposer p ON l.idBungalow = p.idBungalow
                      JOIN Services s ON p.idService = s.idService
                 WHERE nomService = 'Climatisation');
```

Ou

```
SELECT idClient, nomClient
FROM Clients
WHERE villeClient = 'Montpellier'
AND idClient IN (SELECT idClient
                 FROM Locations
                 WHERE idBungalow IN (SELECT idBungalow
                                      FROM Proposer
                                      WHERE idService IN (SELECT idService
                                                          FROM Services
                                                          WHERE nomService =
                                                                   'Climatisation')));
```

Insister sur le fait que ici, il n'est pas possible de remplacer les IN par des =

R04 : le nom et le prénom du chef de l'employé John Deuf.

```
SELECT nomEmploye, prenomEmploye
FROM Employes
WHERE idEmploye IN (SELECT idEmployeChef
                    FROM Employes
                    WHERE nomEmploye = 'Deuf'
                    AND prenomEmploye = 'John');
```

Ou

```
SELECT chef.nomEmploye, chef.prenomEmploye
FROM Employes chef
     JOIN Employes sub ON chef.idEmploye = sub.idEmployeChef
WHERE sub.nomEmploye = 'Deuf'
AND sub.prenomEmploye = 'John';
```

Ces deux requêtes sont équivalentes. Mais statistiquement les étudiants qui utilisent la première version ont beaucoup moins de chance de se tromper surtout quand les noms des alias sur la deuxième version ne veulent rien dire.

R05 : le nombre de bungalows que possède le camping ‘Les Flots Bleus’.

```
SELECT COUNT(*) AS "nb Bungalows"
FROM Campings c
     JOIN Bungalows b ON c.idCamping = b.idCamping
WHERE nomCamping = 'Les Flots Bleus';
```

Bien insister sur la différence entre COUNT(*), COUNT(idBungalow), COUNT(nomBungalow) ...
Faire des rappels sur ce que peuvent retourner les fonctions d'ensemble (COUNT, MIN, MAX, ...) et sur quoi elles peuvent être utilisées.

R06 : la superficie moyenne des bungalows du camping ‘Les Flots Bleus’.

```
SELECT AVG(superficieBungalow) AS "Superficie moyenne"
FROM Campings c
     JOIN Bungalows b ON c.idCamping = b.idCamping
WHERE nomCamping = 'Les Flots Bleus';
```

R07 : le nombre de catégories de services.

```
SELECT COUNT(DISTINCT categorieService) AS "nb Categories"
FROM Services;
```

R08 : le nom de l’employé le mieux payé.

```
SELECT nomEmploye
FROM Employes
WHERE salaireEmploye = (SELECT MAX(salaireEmploye)
                        FROM Employes);
```

On peut indiquer aux étudiants qu'on peut mettre ici un = (même si le IN marche aussi). Car même si il y a deux employés qui ont le plus haut salaire, la requête imbriquée ne retournera quand même qu'une seule ligne.

R09 : le nom des clients qui n’ont jamais réalisé de location.

```
SELECT nomClient
FROM Clients
WHERE idClient IN (SELECT idClient
                   FROM Clients
                    MINUS
                   SELECT idClient
                   FROM Locations);
```

Attention, insiter sur le fait que la requête suivante est fausse

```
SELECT nomClient
FROM Clients
MINUS
SELECT nomClient
FROM Clients c
JOIN Locations l ON l.idClient = c.idClient
```

Rappeler comment fonctionnent les opérateurs ensemblistes et rapeler qu'ils font un DISTINCT

Au premier semestre, certains groupes de TD ont également vu le NOT IN. Si ce n'est pas le cas dans votre groupe de TD, ce n'est pas la peine d'insister car nous reverrons ça plus en détail au second semestre.

```
SELECT nomClient
FROM Clients
WHERE idClient NOT IN (SELECT idClient
                       FROM Locations);
```

R10 : le nom de bungalow et l'identifiant du camping des bungalows qui proposent à la fois le service 'Climatisation' et le service 'TV'.

```
SELECT nomBungalow, idCamping
FROM Bungalows
WHERE idBungalow IN (SELECT idBungalow
                     FROM Services s
                          JOIN Proposer p ON s.idService = p.idService
                     WHERE nomService = 'Climatisation'
                      INTERSECT
                     SELECT idBungalow
                     FROM Services s
                          JOIN Proposer p ON s.idService = p.idService
                     WHERE nomService = 'TV');
```

```
Ou (cette solution ne marcherait pas si on enlevait les idCamping des select mais avec
eux cela va car deux bungalows ne peuvent pas avoir le même nom dans un camping donné)
```

```
SELECT nomBungalow, idCamping
FROM Bungalows b
     JOIN Proposer p ON b.idBungalow = p.idBungalow
     JOIN Services s ON s.idService = p.idService
WHERE nomService = 'Climatisation'
 INTERSECT
SELECT nomBungalow, idCamping
FROM Bungalows b
     JOIN Proposer p ON b.idBungalow = p.idBungalow
     JOIN Services s ON s.idService = p.idService
WHERE nomService = 'TV';
```