Informatique du Tronc commun

TD n°1 Les bases de données – *Corrigé*

Parcours possibles

♪ Si vous avez des difficultés sur ce chapitre : exercices n°3, n°4, n°5, n°1.

♪ ♪ Si vous vous sentez moyennement à l'aise, mais pas en difficulté : exercices n°4, n°5, n°6, n°2.

♪ ♪ ♪ Si vous êtes à l'aise : exercices n°6, n°7, n°2.

I Modèle Entité-association

Exercice n°1 Publications scientifiques

Une base de données concernant des publications scientifiques doit être constituée. Pour modéliser la situation, trois types d'entité sont considérés : `Scientifique, Article` et `Revue`.
Un scientifique peut écrire un ou plusieurs articles. Un article peut être écrit par un ou plusieurs scientifiques et paraître dans une ou plusieurs revues.
Les données enregistrées sont le nom, le prénom et la date de naissance du scientifique, le titre et la date de fin d'écriture de l'article (ou date d'envoi aux différentes revues pour demander une publication), la date de parution dans une revue, le nom de la revue.

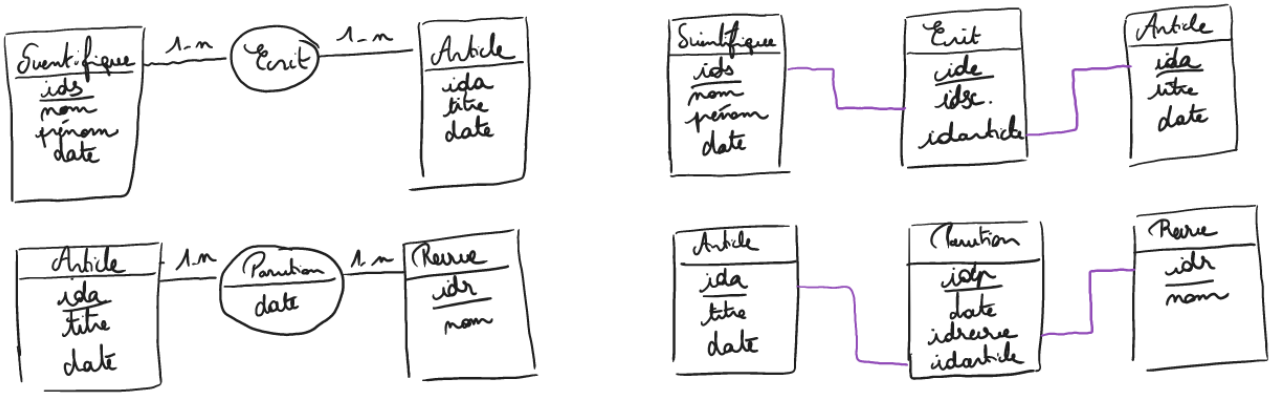
R1. Décrire les types d'associations entre les types d'entités en donnant les cardinalités.

Solution: Entre `Scientifique` et `Article` il y a un type d'association `Écrit` (ou `A écrit`). Un scientifique peut écrire un ou plusieurs articles, donc la cardinalité entre `Scientifique` et `Écrit` est `1-n` . Un article peut être écrit par un ou plusieurs scientifiques, donc la cardinalité entre `Article` et `Écrit par` est `1-n`. Il s'agit d'une association `*-*`

Entre `Article` et `Revue` il y a un type d'association `Paraître` (ou `Publié par`). Un article peut paraître dans une ou plusieurs revues, donc la cardinalité entre Article et `Publié par` est `1-n`. Une revue peut plusieurs un ou plusieurs articles, donc la cardinalité entre `Paraître` et `Revue` est `1-n`. L'association entre `Article` et `Revue` est également de type `*-*`

R2. En déduire les tables utilisées dans le modèle relationnel. Préciser les schémas de relation avec les attributs, les clés primaires et étrangères.

Solution:



L'association entre `Scientifique` et `Article` doit être décomposée en deux associations `1-*` ce qui produit trois tables : `Scientifique`, `Article` et `Écrit`

L'association entre `Article` et `Revue` doit être décomposée en deux associations `1-*` ce qui produit trois tables : `Article`, `Revue` et `Parution`

Schémas de relations :
`Scientifique (ids,nom,prénom,naissance)`
`Article (ida,titre,date_ecrit)`
`Écrit (ide,idarticle,idscientifique)`
`Revue (idr,nom)`
`Parution (idp,idrevue,idarticle,date)`
`idarticle` et `idscientifique` de la table `Écrit` sont les clés étrangères en référence avec les clés primaires **ida** et `ids` des tables `Article` et `Scientifique`.
`idrevue` et `idarticle` sont les clés étrangères de la table `Parution` sont les clés étrangères en référence avec les clés primaires **idr** et `ida` des tables `Revue` et `Article`.

Exercice n°2 Bibliothèque

Une bibliothèque d'un établissement scolaire doit être gérée à l'aide d'une base de données. Pour concevoir un modèle, on considère deux types d'entité : `Livre` et `Élève`.
Le type d'entité `Livre` a pour attributs `Id, Titre, Auteur`.
Le type d'entité `Élève` a pour attributs `Id, Nom, Prénom, Classe`.
On considère un type d'association nommé `Emprunte` entre `Livre` et `Élève`. Un élève peut emprunter aucun, un ou plusieurs livres. Un livre est emprunté par un unique élève.

R1. Préciser le type d'association `1-1`, `1-*` ou `*-*`.

Solution: Le type d'association est `1-*`, qui donne donc deux tables dans le schéma relationnel, en ajoutant une clé étrangère dans la table **Livre** en relation avec la clé primaire `Id` de la table `Élève`.

R2. Définir le schéma relationnel en précisant pour chaque table la clé primaire et d'éventuelles clés étrangères.

Solution:
`Élève (Id, Nom, Prénom, Classe`
`Livre (Id, Titre, Auteur, idélève)`, avec `idélève` une clé étrangère en référence avec la clé primaire `Id` de la table `Élève`, qui fait la relation entre les deux tables.

R3. Peut-on enregistrer deux élèves qui ont le même nom et le même prénom dans la table `Élève` ?

Solution: Il est possible d'enregistrer deux élèves qui ont le même nom et le même prénom, ils le seront avec deux valeurs de `Id` différentes, qui est la clé primaire qui permet d'identifier deux élèves.

On suppose maintenant qu'un livre est disponible en plusieurs exemplaires, donc peut être emprunté par un ou plusieurs élèves en même temps.

R4. Décrire les modifications concernant le modèle entité-association.

Solution: Le type d'association est maintenant `*-*`.

R5. Définir le schéma relationnel en précisant pour chaque table la clé primaire et les éventuelles clés étrangères.

Solution: Il faut maintenant trois tables. On peut ajouter une table `Emprunt` avec des clés étrangères qui vont faire le lien entre la table `Livre` et la table `Élève`
`Élève (Id, Nom, Prénom, Classe`
`Livre (Id, Titre, Auteur)`
`Emprunt(Id, Idélève, Idlivre, date_emprunt,date_retour)`

Un auteur peut avoir écrit plusieurs livres présents dans la base et on souhaite donc éviter la redondance d'informations en n'enregistrant pas plusieurs fois le même nom dans la table `Livre`.

R6. Comment peut-on compléter/modifier le schéma relationnel ?

Solution: On peut ajouter une relation `Auteur` :`Auteur (Id,Nom, Prénom)`
`Élève (Id, Nom, Prénom, Classe`
Et on ajoute dans la table `Livre`, une clé étrangère `idauteur`, en relation avec la clé primaire `Id` de la table `Auteur`. `Livre (Id, Titre, idauteur)`
`Emprunt(Id, Idélève, Idlivre, date_emprunt,date_retour)`

Un livre peut avoir été écrit en commun par plusieurs auteurs.

R7. Comment peut-on compléter/modifier le schéma relationnel ?

Solution: L'association entre `Livre` et `Auteur` est maintenant de type `*-*`, il faut donc ajouter une table pour gérer cela : `Écrit (Id,Idlivre,Idauteur)`, avec `Idlivre` et `Idauteur` des clés étrangères en référence aux clés primaires `Id` des tables `Livre` et `Auteur`.

II Exercices d'applications du cours

Exercice n°3 Requêtes simples sur une seule table ♪

On étudie la table des compositeurs et compositrices dont un extrait est donné ci-dessous.

id	nom	prenom	ddn	ddm	nationalité
1	Lizt	Franz			française
2	Mozart	Wolfang Amadeus			autrichienne
3	van Beethoven	Ludwig	17791215	18670326	allemande
...	...	...	...	...	...
6	Vivaldi	Antonio			italienne
...	...	...	...	...	...

Table des compositeurs (`Compositeurs`)

R1. Est-ce que `nom` peut servir de clé pour la table `Compositeurs` ? Pourquoi ? Proposer une clé de cette table.

Écrire les requêtes en langage SQL qui renvoient :

R2. La liste des noms différents des compositeurs et compositrices.

Solution:

```
SELECT DISTINCT nom
FROM Compositeurs
```

R3. La liste des prénoms et noms des compositeurs et compositrices né.e.s après le 1er janvier 1800.

Solution:

```
SELECT nom, prenom
FROM Compositeurs
WHERE ddn>=18000101
```

R4. La liste des prénoms et noms des compositeurs et compositrices né.e.s au XVIIe siècle, et de nationalité française, les classer par ordre alphabétique de nom puis de prénom.

Solution:

```
SELECT nom, prenom
FROM Compositeurs
WHERE nationalité="française"
ORDER BY  nom, prenom
```

R5. La liste des prénoms et noms des dix compositeurs et compositrices les plus ancien.ne.s renseigné.e.s dans la table.

Solution:

```
SELECT nom, prenom
FROM Compositeurs
WHERE nationalité="française"
ORDER BY  ddn
LIMIT 10
```

Exercice n°4 Requêtes sur deux (ou plus) tables ♪

On étudie la base de données suivante, constituée de 2 tables, celle de l'exercice précédent et celle ci-dessous.

id	titre	annee	periode	forme	duree	idcompo
1	La flûte enchantée	1791	classique	opéra	165	2
2	Neuvième Symphonie	1824	romantique	symphonie	70	3
3	Les quatre saisons	1725	baroque	concerto	40	6
...	...	...	...	...	...	...

Table des Œuvres (`Oeuvres`).

R1. Identifier les clés primaires et clés étrangères de chaque table.

Solution:

Écrire les requêtes en langage SQL qui renvoient :

R2. La liste des titres de œuvres composées par Cécile Chaminade qui dure moins d'une heure.

Solution:

```
SELECT titre
FROM Oeuvres
JOIN Compositeurs AS C
ON idcompo=C.id
WHERE nom = "Chaminade" AND prenom="Cécile" AND duree<60
```

R3. La liste des titres des œuvres composées par une compositrice ou un compositeur allemand.e au cours du XIX^{e} siècle. On renverra également les noms et prénoms des compositrices et compositeurs.

Solution:

```
SELECT nom, prenom, titre
FROM Oeuvres
JOIN Compositeurs AS C
ON idcompo=C.id
WHERE annee>=1800 AND annee<=1899 AND nationalite="allemande"
```

R4. La liste des noms et prénoms des compositrices et compositeurs ayant composé au moins une symphonie.

Solution:

```
SELECT nom, prenom
FROM Oeuvres
JOIN Compositeurs AS C
ON idcompo=C.id
WHERE forme="symphonie"
```

R5. ♪ ♪ Le titre de l'œuvre de durée maximale, et son compositeur ou sa compositrice.

Solution:

```sql
SELECT nom, prenom, titre
FROM Oeuvres
JOIN Compositeurs AS C
ON idcompo=C.id
ORDER BY duree DESC -- ordre décroissant de durée
LIMIT 1 -- une seule oeuvre de durée maximale
```

R6. ♪♪ Le titre de la deuxième œuvre la plus longue, avec le nom de son compositeur ou de sa compositrice.

Solution:

```sql
SELECT nom, prenom, titre
FROM Oeuvres
JOIN Compositeurs AS C
ON idcompo=C.id
ORDER BY duree DESC -- ordre décroissant de durée
LIMIT 1 -- une seule oeuvre
OFFSET 1 -- on commence à la ligne 1 (2è ligne du tableau)
```

Exercice n°5 Fonctions d'agrégation ♪

On étudie la même base de données que dans l'exercice précédent. Écrire les requêtes en langage SQL qui renvoient :

R1. La durée moyenne des œuvres renseignées.

Solution:

```sql
SELECT AVG(duree)
FROM Oeuvres
```

R2. Le nombre de compositeurs et compositrices renseigné.e.s dans la base.

Solution:

```sql
SELECT COUNT(id)
FROM Compositeurs
```

R3. Le nombre de nationalités différentes renseignées dans la base.

Solution:

```sql
SELECT COUNT(DISTINCT nationalite)
FROM Compositeurs
```

R4. La durée totale des œuvres composées présentes dans la table.

Solution:

```sql
SELECT SUM(duree)
FROM Oeuvres
```

R5. La durée totale des œuvres composées par chaque compositeur ou compositrice, avec leurs noms et prénoms.

Solution:

```sql
SELECT nom, prenom, SUM(duree)
FROM Oeuvres
JOIN Compositeurs AS C
ON C.id=idcompo
GROUP BY idcompo
```

R6. La durée moyenne par forme des œuvres renseignées (on fera apparaître deux colonnes, classée par durée décroissante : forme et « durée moyenne »).

Solution:

```sql
SELECT forme, AVG(duree)
FROM Oeuvres
GROUP BY forme
ORDER BY AVG(duree) DESC
```

R7. ♪ ♪ Les titres d'œuvres composées par au moins trois compositeurs ou compositrices différent.e.s.

Solution:

```sql
SELECT titre
FROM Oeuvres
GROUP BY titre
HAVING COUNT(DISTINCT idcompo)>=3
```

R8. ♪ ♪ ♪ La liste des noms et prénoms de compositeurs ou compositrices ayant composé au moins 9 symphonies.

Solution:

```sql
SELECT nom, prenom
FROM Oeuvres AS O
JOIN Compositeurs AS C
ON C.id = idcompo
WHERE forme = "symphonie"
GROUP BY idcompo
HAVING COUNT(O.id)>=9
```

III Exercices d'approfondissements

Exercice n°6 Pays du monde ♪ ♪

On reprend la base de données du cours.

R1. Rédiger une requête SQL donnant le nom des dix pays asiatiques les plus grands.

Solution:

```sql
SELECT name
FROM country
```

```
WHERE Continent = "Asia"
ORDER BY Population DESC
LIMIT 10
```

R2. Rédiger une requête SQL donnant le nom et la date d'indépendance des dix pays les plus anciens.

Solution:

```
SELECT name, IndepYear
FROM country
ORDER BY IndepYear ASC
LIMIT 10
```

R3. Rédiger une requête SQL donnant la liste des langues non officielles pratiquées en France, par ordre décroissant d'importance.

Solution:

```
SELECT Language
FROM country
JOIN countrylanguage
ON  CountryCode = Code
WHERE IsOfficial = "F" AND Name = "France"
ORDER BY Percentage DESC
```

R4. Rédiger une requête SQL donnant le nom des cinq pays européens les moins densément peuplés.

Solution:

```
SELECT Name
FROM country
WHERE Continent = "Europe"
ORDER BY Population/SurfaceArea
LIMIT 5
```

R5. Rédiger une requête SQL donnant la liste des pays ayant adopté le Français parmi leurs langues officielles.

Solution:

```
SELECT Language
FROM country
JOIN countrylanguage
ON  CountryCode = Code
WHERE IsOfficial = "T" AND Language = "French"
```

R6. Rédiger une requête SQL donnant la liste des districts (attribut de la table `country` nommé `Region`) des États-Unis dont la population totale est supérieure à 3 000 000.

Solution:

```sql
SELECT Region
FROM country
WHERE Name = "USA"
GROUP BY Region
HAVING SUM(Population) > 3e6
```

R7. Rédiger une requête SQL donnant les cinq villes les plus peuplées d'Europe.

Solution:

```sql
SELECT V.Name
FROM city AS V
JOIN country AS P
ON CountryCode = Code
WHERE Continent = "Europe"
ORDER BY V.Population DESC
LIMIT 5
```

R8. ♪ ♪ Rédiger une requête SQL donnant les cinq villes les plus peuplées d'Europe parmi celles qui ne sont pas des capitales.

Solution:

```sql
SELECT V.Name
FROM city AS V
JOIN country AS P
ON CountryCode = Code
WHERE Continent = "Europe"
ORDER BY V.Population DESC
LIMIT 5
EXCEPT -- on enlève les capitales
SELECT V.Name
FROM city AS V
JOIN country AS P
ON Capital = id -- sélection des capitales
```

R9. ♪ ♪ Rédiger une requête SQL donnant les capitales des pays dans lesquels l'allemand est langue officielle.

Solution:

```sql
SELECT V.Name
FROM city AS V
JOIN country AS P
JOIN countrylanguage AS L
ON id = Capital AND L.CountryCode = Code
WHERE Language = "German" AND IsOfficial = "T"
```

R10. ♪ ♪ Rédiger une requête SQL déterminant les cinq langues les plus parlées au monde.

Solution:

```sql
SELECT Language
FROM country AS P
JOIN countrylanguage AS L
GROUP BY Language
ORDER BY SUM(Percentage * Population) DESC
LIMIT 5
```

R11. ♪ ♪ ♪ Donner la liste des pays dont l'espérance de vie est supérieure ou égale à celle de la France.

Solution:

```sql
SELECT Name
FROM country
WHERE LifeExpectancy > (SELECT LifeExpectancy
                        FROM country
                        WHERE Name = "France")
```

R12. ♪ ♪ ♪ Donner le pourcentage de pays dont le PNB est supérieur ou égal à la moyenne.

Solution:

```sql
SELECT 100 * COUNT(Code) / ( SELECT COUNT(Code) FROM country )
FROM country
WHERE GNP > ( SELECT AVG(GNP) FROM country )
```

R13. ♪ ♪ ♪ Rédiger une requête SQL déterminant, pour chaque continent, le pays le plus peuplé.

Solution:

```sql
SELECT Continent, Name, population
FROM country
WHERE (continent , population)  -- on récupère les pays dont le
   continent et la population = au continent et à sa population
   maximale
IN ( SELECT continent,MAX(population)
     FROM country
     GROUP BY Continent) -- sous-requête qui récupère la population
   maximale pour chaque continent
```

Exercice n°7 Locations ♪ ♪

Le lycée Camille Vernet propose à ses étudiant.e.s la location entre étudiant.e.s de quatre types de vélos :

1. les vélos de route,
2. les VTT,
3. les vélos à assistance électrique,
4. les vélos cargos,

L'adhésion au système de prêt est conditionnée par la mise à disposition d'au moins un vélo de manière permanente. La location se fait à la semaine (numérotée de 1 à 52) et chaque année scolaire la base de données est réinitialisée. Les requêtes ne valent donc que pour l'année en cours.

Les personnes sont enregistrées dans la table `Proprietaire`.

Les vélos sont répertoriés dans la table `Velo`. Chaque vélo y est enregistré avec exactement un propriétaire et un prix de référence pour la semaine de location.

Un vélo peut être en relation avec au plus 52 locataires par an via la table `Location` où `idSemaine` (un entier choisi entre 1 et 52) indique la semaine de location du vélo `idVelo` par le locataire `idLoc` (clé étrangère en relation avec `idProp` de la table `Proprietaire`) et où `kilometrage` est un entier qui n'est renseigné qu'une fois que la prestation est payée par le locataire au propriétaire du vélo loué.

Ainsi, avec la définition des clefs primaires un même locataire peut emprunter plusieurs véhicules la même semaine mais un véhicule a zéro ou un locataire par semaine.

Location
idloc
idSemaine
idVelo
idLoc
kilometrage

Velo
idVelo
TypeVelo
idProp
Prix
TelContact

Proprietataire
idProp
nom
prenom
TelContact

R1. Écrire la requête SQL qui donne la liste des types de vélos sans doublon.

Solution:

```sql
SELECT DISTINCT TypeVelo
FROM Velo
```

R2. Écrire la requête SQL qui donne le nombre de vélos.

Solution:

```sql
SELECT COUNT(idVelo)
FROM Velo
```

R3. Écrire la requête SQL qui donne le nombre total de kilomètres parcourus.

Solution:

```sql
SELECT SUM(kilometrage)
FROM Location
```

R4. Écrire la requête SQL qui donne le prix maximum.

Solution:

```sql
SELECT MAX(Prix)
FROM Velo
```

R5. Écrire la requête SQL qui donne le nom, prénom du propriétaire et le type de vélo dont le prix est minimal.

Solution:

```sql
SELECT nom, prenom, TypeVelo
FROM Velo AS V
JOIN Proprietaire AS P
ON V.idProp = P.idProp
ORDER BY Prix ASC
```

```
LIMIT 1
```

R6. Écrire la requête SQL qui donne le nom, prénom des propriétaire et le type de vélos des trois vélos les plus chers.

Solution:

```
SELECT nom, prenom, TypeVelo
FROM Velo AS V
JOIN Proprietaire AS P
ON V.idProp = P.idProp
ORDER BY Prix DESC
LIMIT 3
```

R7. Écrire la requête SQL qui donne le prix moyen de location de vélos par types de vélo, classé par ordre de prix croissant.

Solution:

```
SELECT TypeVelo , AVG(Prix)
FROM Velo
GROUP BY TypeVelo
ORDER BY AVG(Prix) ASC
```

R8. Écrire la requête SQL qui donne la liste des noms et prénoms des propriétaires et du nombre de vélos qu'elles.ils mettent à disposition.

Solution:

```
SELECT nom, prenom, COUNT(idVelo)
FROM Velo AS V
JOIN Proprietaire AS P
ON V.idProp = P.idProp
```

R9. ♪ ♪ Écrire la requête SQL qui donne la liste noms et prénoms des propriétaires mettant au moins trois vélos à disposition.

Solution:

```
SELECT nom, prenom, COUNT(idVelo)
FROM Velo AS V
JOIN Proprietaire AS P
ON V.idProp = P.idProp
HAVING COUNT(idVelo)>=3
```

R10. ♪ ♪ Pour chaque propriétaire qui en propose au moins un à louer, référencé par son numéro, indiquer le nombre de vélos à assistance électrique qu'elle.il offre à la location. On rangera les réponses par ordre décroissant du nombre de vélos puis (à nombre de vélos égal) par numéro de propriétaire croissant.

Solution:

```sql
SELECT idProp, COUNT(*) AS nbVAE
FROM Velo
WHRE TypeVelo = 3
GROUP BY idProp
ORDER BY nbVAE, idProp
```

R11. ♪ ♪ Pour chaque propriétaire, référencé par son numéro, afficher la liste des semaines où il a loué au moins un vélo cargo. On rangera les réponses pour chacun.e. des propriétaires par ordre croissant de ses semaines de locations.

Solution:

```sql
SELECT idLoc, idSemaine
FROM Velo AS V
JOIN Localtion AS L
ON V.idVelo = L.idVelo
WHERE TypeVelo=4
ORDER BY idLoc, idSemaine
```

R12. ♪ ♪ ♪ Écrire la requête SQL qui donne la liste des vélos à assistance électrique, noms et prénoms de leurs propriétaires, non encore réservés pour les semaines 50, 51 et 52.

Solution:

```sql
SELECT nom, prenom, idVelo
FROM Velo AS V
JOIN Proprietaire AS P
JOIN Location AS L
ON V.idProp = P.idProp AND L.idVelo = V.idVelo
WHERE TypeVelo=3 AND idSemaine NOT IN (50,51,52) -- pas réservé ces
   semaines-là
```

R13. ♪ ♪ ♪ Pour chaque propriétaire, référencé et classé de manière croissante par son numéro, lister les vélos qui lui ont été loués en permanence les 10 premières semaines de l'année en cours et lui ont rapporté sur cette période plus de 200 euros. Pour chaque propriétaire, on classera les vélos par type de vélo classés par ordre alphabétique puis, au sein d'un même type pour un même propriétaire par numéro décroissant.

Solution:

```sql
SELECT idProp, idVelo
FROM Velo AS V
JOIN Location AS L
ON V.idVelo = L.idVelo
WHERE idSemaine<=10
GROUP BY idProp
HAVING COUNT(idSemaine)=10 AND SUM(Prix)>200 --Loué en permanence =
   pendant 10 semaine
ORDER BY idProp ASC, TypeVelo ASC, idVelo DESC
```

R14. ♪ ♪ ♪ Pour chaque propriétaire, référencé et classé de manière croissante par son numéro, lister les numéros des vélos qu'il possède et qui ont déjà été utilisés pour plus de 700 km en une semaine mais qui n'ont été réservés ou loués qu'au plus par trois locataires différents depuis le début de l'année.

Solution:

```
SELECT idProp, idVelo
FROM Velo AS V
JOIN Location AS L
ON V.idVelo = L.idVelo
WHERE kilometrage > 700
AND idVelo IN ( SELECT idVelo
                FROM Location AS L
                GROUP BY idVelo
                HAVING COUNT(DISTINCT idLoc)<=3 )
ORDER BY idProp
```

R15. ♪ ♪ ♪ Pour chaque vélo loué deux semaines consécutives au même locataire, lister les locataires référencé.e.s par leur numéro, qui ont doublé (ou plus) leur kilométrage d'une semaine à la suivante.

On indiquera les kilométrages respectifs. Pour un vélo donné, les locataires seront classé.e.s du plus sportif.ve sur la deuxième semaine considérée au moins sportif.ve. Les vélos sont eux à ranger par ordre croissant de numéro sans autre indication.

Solution:

```
SELECT L1.idVelo, L2.idLoc, L1.kilometrage, L2.kilometrage
FROM Location AS L1
JOIN Location AS L2
ON L1.idVelo = L2.idVelo AND L1.idLoc=L2.idLoc
WHERE L2.kilometrage >= 2*L1.kilometrage AND L2.idSemaine = L1.
   idSemaine + 1
ORDER BY L2.idVelo ASC, L2.kilometrage DESC
```