Lycée Lakanal

2025 - 2026

PC

TD 1 – Bases de données

Correction

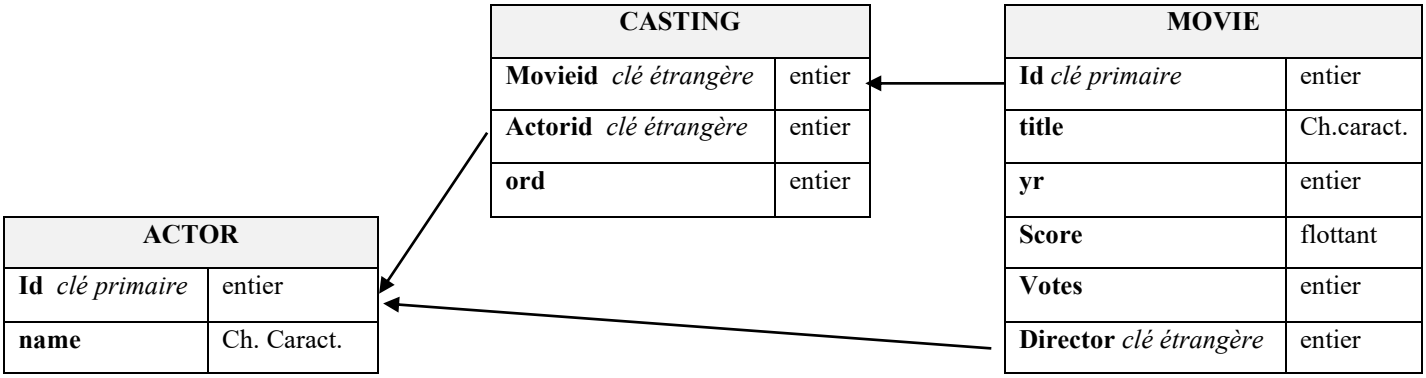
Démarrage

Sélectionner "Fichier", "ouvrir" (ou CTRL+O), et indiquer le dossier où trouver votre base de données, à savoir la zone d'échange de la classe, dossier informatique, puis bases_de_donnees et enfin bdd.

Sélectionnez alors la base de données movies.

Exercice 1

Parcourir les différentes tables de cette base de données, et dresser son schéma relationnel.



Dans les exercices 2 et 3, on travaillera avec la base de données France, que l'on ouvrira suivant la même procédure que ci-dessus. Nous n'utiliserons que sa table Villes2, dont le schéma relationnel est le suivant :

Villes2

Attribut	Type
Numéro	Entier
Dept	Chaîne de caractères *merci la Corse* !
Nom	Chaîne de caractères
Code postal	Chaîne de caractère (*pourquoi* ?)
Pop_2010	Entier
densité	Réel

Clé primaire

Exercice 2

On répondra aux questions ci-dessous en écrivant des « requêtes » (c'est-à-dire des instructions), sur le modèle de celles données en annexe. On doit répondre à chaque question ou sous-question grâce à une seule et unique requête.

1. Y a – t – il des villes de densité nulle ? Sont – ce les mêmes que celles qui avaient une population nulle en 2010 ? Explication ?

```
SELECT Nom, Pop_2010, Densité_2010 FROM Villes2
  WHERE (Densité_2010 = 0 OR  Pop_2010 = 0 ) AND NOT (Densité_2010 = 0 AND  Pop_2010 = 0 )
```

On reçoit pour réponse :

	Nom	Pop_2010	Densité_2010
1	Blieux	55	0
2	Majastres	2	0
3	Gestiès	9	0
4	La Fajolle	15	0
5	Fontanès-de-Sault	5	0
6	Rochefourchat	1	0
7	La Bâtie-des-Fonds	8	0

Les densités ont probablement été arrondies.

2. Déterminer le nombre de villes pour chaque département.

```
SELECT COUNT(*), Dept FROM Villes2 GROUP BY Dept
```

3. Donner la population de chaque département.

```
SELECT SUM(Pop_2010), Dept FROM Villes2 GROUP BY Dept
```

4. Donner une estimation de la superficie de la ville de Paris (vous devez trouver environ $105\ km^2$).

```
SELECT Pop_2010/Densité_2010 FROM Villes2 WHERE Nom= 'Paris'
```

5. Sortir les données de la ville (ou des villes) de densité (non nulle) minimale.

Solution de feignant : c'est laid, incomplet, mais ça donne une réponse partielle

```
SELECT * FROM Villes2 WHERE densité_2010>0 ORDER BY densité_2010 LIMIT 1
```

Mieux, même si ça semble plus compliqué :

```
SELECT * FROM VILLES2
WHERE Densité_2010=(SELECT MIN(Densité_2010) FROM Villes2 WHERE Densité_2010>0)
```

6. Donnner la liste des noms et la population 2010 des villes dont le nom commence par B ou dont la population 2010 était supérieure à 50 000.

```
SELECT Nom , Pop_2010 FROM villes2
WHERE Nom LIKE 'B%' OR POP_2010>50000
```

Exercice 3

1. Que fait la stupide requête suivante ? Qu'aurait – on pu écrire d'autre, avec le même résultat ?

```
SELECT Nom , Pop_2010 FROM villes2
WHERE Nom IN
(SELECT Nom FROM villes2
WHERE Nom LIKE 'B%')
```

Cette requête renvoie le nom et la population des villes dont le nom commence par la lettre B. La requête suivante aurait été plus logique :

```
SELECT Nom , Pop_2010 FROM villes2 WHERE Nom LIKE 'B%'
```

2. Donner la liste des villes dont la population est supérieure (ou égale) à celle de toutes les villes dont le nom commence par A .

```
SELECT Nom  FROM villes2 WHERE Pop_2010 >=
(SELECT MAX(POP_2010) FROME Villes2 WHERE NOM LIKE 'A%')
```

3. On souhaite déterminer la population moyenne (en 2010) de l'ensemble des villes dont le nom commence par 'A', département par département.

Que renvoient les requêtes suivantes ?

```
SELECT AVG(nb) FROM
(SELECT Dept, Nom, COUNT(*) as nb
FROM villes2
WHERE Nom LIKE 'A%'
GROUP BY Dept)
As result
```

(*on reçoit pour réponse :* 19,77)

On obtient le nombre moyen par département de villes dont le nom commence par A.

```
SELECT AVG(nb) FROM
(SELECT Dept, Nom, COUNT(Pop_2010) as nb
FROM villes2
GROUP BY Dept)
As result
```

(*on reçoit pour réponse :* 359,8039)

On obtient le nombre moyen de villes par département. Rq : COUNT(Pop_2010) renvoie la même chose que COUNT()

```
SELECT nb FROM
(SELECT Dept, Nom, AVG(Pop_2010) as nb
FROM villes2
WHERE Nom Like ('A%')
GROUP BY Dept)
As result
```

(*on reçoit pour réponse :* 1637,19)

Cette requête renvoie le résultat voulu.

4. Que fait la requête suivante ?

```
SELECT Nom , Pop_2010 FROM villes2
WHERE Nom IN
(SELECT Nom FROM villes2
WHERE Pop_2010 >50000)
```

On pourra comparer son résultat avec celui de l'instruction suivante :

```
SELECT Nom , Pop_2010 FROM villes2
WHERE Pop_2010 >50000
```

La première requête renvoie la liste des villes qui ont le même nom qu'un ville de plus de 50 000 habitants, avec leur popolation. A tritre d'exemple, une ville nommée Saint-Denis et ayant 350 habitants figurerait dans cette liste, puisqu'il existe une autre ville, nommée Saint Denis également, et ayant plus de 50 000 habitants.

Exercice 4

On s'occupe ici de la base de données mondial, située au même endroit que la base précédente. Cette base de données contient de nombreuses tables de données géographiques. Parmi celles – ci, on trouve une table nommée Country, qui possède 6 attributs :

Name Code Capital Province Area Population.

Les quatre premiers sont des chaînes de caractères, le cinquième un nombre flottant et le sixième un entier. Le code du pays est une clé primaire de la table, son unicité est donc garantie.

1. Rédiger une requête SQL permettant d'obtenir la liste des noms de pays dont la population excède 60 000 000 d'habitants.

```
SELECT name FROM country WHERE population>60000000
```

2. Rédiger une requête SQL permettant d'obtenir la même liste, triée par ordre alphabétique.

```
SELECT name FROM country WHERE population>60000000 ORDER BY Name
```

3. Rédiger une requête SQL permettant d'obtenir la même liste, triée par ordre décroissant de population.

```
SELECT name FROM country WHERE population>60000000 ORDER BY Pop_2010 DESC
```

4. Rédiger une requête SQL permettant d'obtenir la liste des noms dix plus petits pays (en terme de surface).

```
SELECT name FROM country ORDER BY Area limit 10
```

5. Rédiger une requête SQL permettant d'obtenir la liste des quinze suivants.

```
SELECT name FROM country ORDER BY area DESC LIMIT 15 OFFSET 10
```

6. En utilisant les tables Country et Island, dresser la liste des noms de pays ou d'iles (*une seule liste,merci !*).

```
SELECT name FROM Country UNION SELECT Name FROM Island
```

7. Dresser la liste des noms des pays ayant le même nom qu'une île.

```
SELECT name FROM Country INTERSECTION SELECT Name FROM Island
```

Exercice 5

On revient à la base mondial. On y on trouve une table nommée encompasses possédant trois attributs :

Country Continent Percentage.

Le premier attribut est le code du pays, le deuxième le nom du continent, et le dernier la portion du pays présente sur le continent. La clé primaire de cette table est le couple (Country, Continent), et la valeur du troisième argument ne peut pas être nulle. Cette seconde table possède un attribut en commun avec la première table : l'attribut Country de la table encompasses renvoie en effet à l'attribut Code de la table country (c'est une clé étrangère pour la table encompasses). Ceci va nous permettre de croiser les informations de ces deux tables.

Q1. Rédiger une requête SQL permettant d'obtenir la liste des pays dont une partie au moins du territoire est en Europe.

```
SELECT Name FROM country JOIN encompasses ON country=code AND continent='Europe'
```

Q2. Rédiger une requête SQL permettant d'obtenir le nombre de pays qui sont à cheval sur plusieurs continents.

Voici plusieurs possibilités :

```
SELECT DISTINCT country.name
  FROM country JOIN encompasses
  On country.code=encompasses.country
  WHERE encompasses.percentage<100
```

```
SELECT name
  FROM country JOIN encompasses
 On country.code=encompasses.country
 WHERE encompasses.percentage<100
 GROUP BY name
```

La requête ci – dessous ne fait pas tout à fait ce que l'on veut….

```
SELECT count(*) C, name
  FROM country JOIN encompasses
```

```
On country.code=encompasses.country
GROUP BY name
ORDER BY C DESC
```

… mais si on creuse un peu plus l'idée, cela donne aussi le résultat attendu :

```
SELECT N FROM
  (SELECT count(*) C, name N
  FROM country JOIN encompasses
  On country.code=encompasses.country
  GROUP BY N)
WHERE C>1
```

Q3. Rédiger une requête SQL permettant d'obtenir (super intéressant) le nombre moyen d'habitants des pays du continent américain qui comptent moins de dix habitants par kilomètre carré.

```
SELECT AVG (P) FROM
  (SELECT population P
    FROM country JOIN encompasses
      ON country.code=encompasses.country AND continent='America' AND 1.0*Population/Area<10)
```

Si l'on veut la liste de ces pays, et si l'on a peur que deux pays aient le même nom :

```
SELECT DISTINCT country.name FROM country JOIN encompasses
  ON country.code=encompasses.country
WHERE encompasses.Continent='America' AND country.population/country.area < 10
```

ce qui revient encore à :

```
SELECT country.name FROM country JOIN encompasses
  ON country.code=encompasses.country
WHERE encompasses.Continent='America' AND country.population/country.area < 10
GROUP BY country.name
```

Exercice 6

Dans la même base de données figure une table nommée city qui possède les attributs suivants :

Name Country Province Population Longitude Latitude.

L'attribut Coutry est toujours le code du pays : c'est, à nouveau, une clé étrangère pour la table city.

Q1. Déterminer les capitales européennes situées à une latitude supérieure à 60, ainsi que les noms des pays correspondants.

Cela commence à se compliquer… il faut croiser les tables country, city et encompasses :

```
SELECT country.capital, country.name
    FROM country JOIN encompasses
        ON country.code=encompasses.country
    JOIN city
        ON country.code=city.country
    WHERE (city.latitude>60 and encompasses.continent='Europe' and city.name=country.capital)
```

ou, si l'on préfère :

```
SELECT country.capital, country.name
    FROM country JOIN encompasses, city
            ON country.code=encompasses.country AND country.code=city.country
    WHERE (city.latitude>60 and encompasses.continent='Europe' and city.name=country.capital)
```

Que fait la requête (correcte, mais ne répondant pas à la question posée) suivante ?

```
SELECT DISTINCT country.capital
    FROM country JOIN encompasses
    ON country.code=encompasses.country
    JOIN city
    ON country.code=city.country WHERE (city.latitude>60 and encompasses.continent='Europe')
```

Q2. La table Language posssède les attributs suivants :

Country Name Percentage.

L'attribut Country est le code du pays, Name le nom d'une langue parlée dans ce pays, et Percentage le pourcentage d'habitants dont c'est la langue maternelle.

a. Donner la liste ordonnée (dans le sens décroissant), des 10 langues parlées dans le plus de pays.

Une jointure n'est pas nécessaire, puisque toutes les informations nécessaires figurent dans la table Language

```
SELECT name, COUNT(*) N
  FROM language
  GROUP BY name
  ORDER BY N DESC LIMIT 10
```

b. Quelles sont les 5 langues les plus parlées dans le monde ? On précisera pour chacune d'entre elles le nombre de personnes qui la parlent.

Cette fois – ci, il faut croiser les tables country et language :

```
SELECT L.name , SUM( C. population * L.percentage / 100) S
  FROM language L  JOIN country C
  ON  L.country = C.code
  GROUP BY L.name
  ORDER BY S DESC LIMIT 5
```

c. Quel est le pays partiellement francophone dans lequel la langue française est la plus minoritaire ?

```
SELECT country.name , language.percentage
  FROM language  JOIN country
  ON country.code = language.country
  WHERE language.name = 'French'
  AND language.percentage =
    ( SELECT MIN ( percentage ) FROM language WHERE name = 'French' )
```

d. Quelle est le pourcentage de personnes francophones dans le monde ?

```
SELECT 100*A/B FROM
  (   (SELECT SUM( C. population * L.percentage / 100) A
        FROM language L  JOIN country C
        ON  L.country = C.code AND L.name = 'French'),
```

```
  (SELECT SUM( population ) B FROM country)   )
```

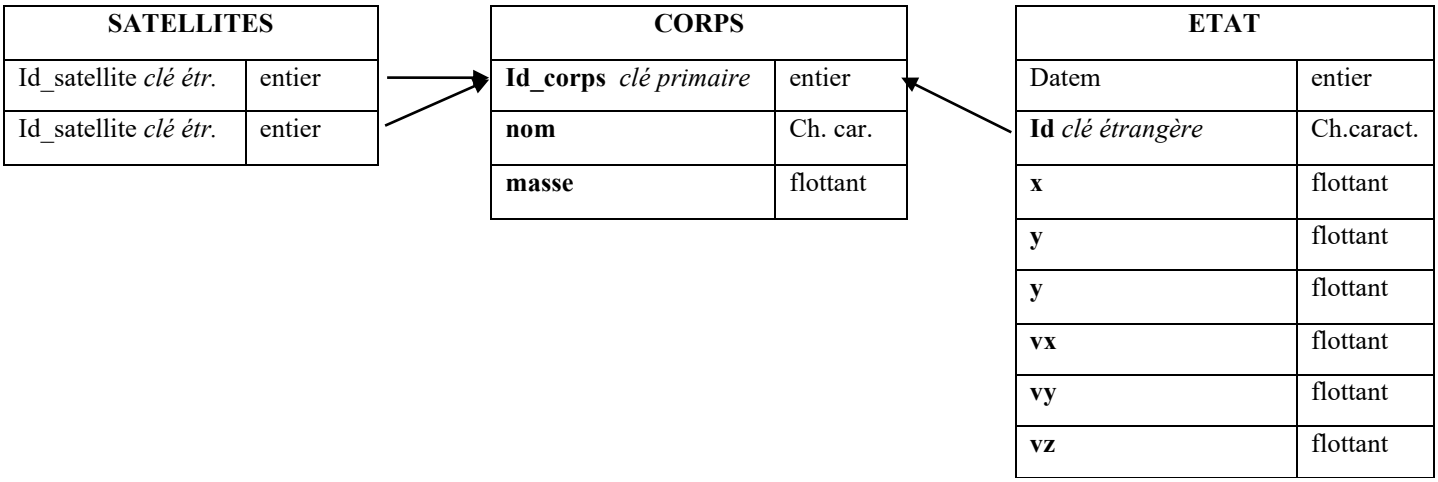
ou encore :

```
SELECT 100*SUM( C. population * L.percentage / 100)/ (SELECT SUM( population ) FROM country)
  FROM language L  JOIN country C
  ON  L.country = C.code AND L.name = 'French'
```

Nous ne travaillerons dans tous les exercices qui suivent qu'avec une seule base de données : la base de données eph2, qui se trouve à l'endroit habituel.

Exercice 7

Parcourir les différentes tables, puis donner le schéma relationnel de cette base de données.



SATELLITES	
Id_satellite *clé étr.*	entier
Id_satellite *clé étr.*	entier

CORPS	
Id_corps *clé primaire*	entier
nom	Ch. car.
masse	flottant

ETAT	
Datem	entier
Id *clé étrangère*	Ch.caract.
x	flottant
y	flottant
y	flottant
vx	flottant
vy	flottant
vz	flottant

Exercice 8

On ordonne les corps par masse décroissante. Parmi les corps suivants : Titan, Terre, Lune, Ganymede, quels sont ceux dont le rang est situé entre la 13ème et la 22ème position ?

```
SELECT nom FROM corps
  WHERE
    nom IN (select nom from corps ORDER BY masse DESC LIMIT 10 OFFSET 12)
    AND
    (nom LIKE 'Lune' OR  nom LIKE 'Terre' OR nom LIKE 'Titan' OR nom LIKE 'Ganymede')
```

Exercice 9

Donner la liste des noms des satellites de Neptune présents dans cette base de données.

```
SELECT nom FROM corps JOIN satellites
  ON id_satellite=id_corps
  AND id_planete=
    (SELECT id_corps FROM corps WHERE nom LIKE 'NEPTUNE')
```

Exercice 10

Deux corps n'ont aucune donnée présente dans la table etat. Quels sont – ils ?

```
SELECT nom FROM corps JOIN etat
  ON corps.id_corps=etat.id_corps
    AND corps.id_corps NOT IN
    (SELECT id_corps FROM etat GROUP BY id_corps)
```

Exercice 11

De quelle(s) planète(s) les deux corps ci – dessus sont – ils satellites ?

```
SELECT nom FROM corps JOIN satellites
  ON id_corps=id_planete
  AND id_satellite NOT IN
    (SELECT corps.id_corps FROM corps JOIN etat
      ON corps.id_corps=etat.id_corps)
  GROUP BY nom
```

Exercice 12

On néglige les deux corps ci – dessus. Quelle est (s'il y en a une), la première date à laquelle tous les corps ont des positions établies dans la table etat ?

```
SELECT MIN(D) FROM
(SELECT DATEM AS D, COUNT() AS C FROM ETAT GROUP BY Datem
  HAVING C =
    (SELECT MAX(E) FROM (SELECT COUNT() AS E FROM ETAT GROUP BY Datem) ) )
```

Exercice 13

Donner le nombre d'entrées enregistrées dans la table etat concernant le système planétaire de Neptune.

```
SELECT SUM(C) FROM
  (SELECT id_corps, COUNT(*) C FROM etat
    WHERE id_corps IN
      (SELECT id_satellite FROM satellites
        WHERE id_planete=(SELECT id_corps FROM corps WHERE NOM LIKE 'Neptune'))
        OR id_corps=(SELECT id_corps FROM corps WHERE NOM LIKE 'Neptune')
   GROUP BY id_corps)
```