

FICHE DE RÉVISION MYSQL

Crash course bilingue FR / EN

Examen : jeudi 18 juin 2026 à 10 h 00 - distanciel

Exam: Thursday 18 June 2026 at 10:00 - remote

Objectif : savoir écrire rapidement une requête correcte, expliquer chaque clause et manipuler la structure d'une base MySQL.

Goal: quickly write a correct query, explain every clause, and manipulate a MySQL database structure.

Priorité spéciale : alias de colonnes et de tables. Jointures limitées aux trois schémas indispensables.

Special priority: column and table aliases. Joins are limited to the three essential patterns.

Contenu du pack

- 1. Cours condensé : LID/DQL, LDD/DDD, fonctions, agrégats, sous-requêtes et contraintes.
1. Condensed course: DQL, DDL, functions, aggregates, subqueries, and constraints.
- 2. Dictionnaire des commandes réellement rencontrées dans les supports.
2. Dictionary of the commands actually found in the course material.
- 3. Exercices progressifs, examen blanc et corrections.
3. Progressive exercises, a mock exam, and solutions.
- 4. Fichier SQL réinitialisable et entraînement HTML dynamique fournis séparément.
4. A resettable SQL file and dynamic HTML trainer are provided separately.

Plan de travail recommandé

Créneau / Slot	Travail / Work	But / Goal
Ce soir - 45 min	Lire les sections 1 à 4. / Read sections 1-4.	SELECT, alias, WHERE, NULL.
Ce soir - 60 min	Faire les exercices Alias + Filtres. / Do Alias + Filter drills.	Automatiser les formes de base. / Make core forms automatic.
Ce soir - 60 min	Agrégats, GROUP BY, HAVING, sous-requêtes. / Aggregates and subqueries.	Partie la plus rentable. / Highest-value section.
Ce soir - 45 min	LDD : CREATE, ALTER, contraintes. / DDL and constraints.	Savoir reconstruire une table. / Rebuild a table.
Demain - 35 min	Examen blanc sans correction. / Mock exam without solutions.	Vitesse et méthode. / Speed and method.
Dernières 15 min	Relire la fiche ultra-courte. / Read the final cram sheet.	Éviter les erreurs mécaniques. / Avoid mechanical errors.

Sommaire / Contents

1. 1. Périmètre de l'examen et priorités / Scope and priorities
2. 2. Anatomie d'une requête SELECT / Anatomy of a SELECT query
3. 3. Alias de colonnes et de tables / Column and table aliases
4. 4. Sélection, DISTINCT, NULL et tris / Selection, DISTINCT, NULL, sorting
5. 5. Filtres et opérateurs / Filters and operators
6. 6. Fonctions numériques, texte, dates et NULL / Numeric, text, date, NULL functions
7. 7. Agrégats, GROUP BY, HAVING et GROUP_CONCAT
8. 8. Sous-requêtes / Subqueries
9. 9. Jointures minimales / Minimal joins
10. 10. LDD/DDL : bases, tables, colonnes et contraintes
11. 11. INSERT, UPDATE, DELETE et copies de tables
12. 12. Modélisation relationnelle : minimum théorique / Relational modelling essentials
13. 13. Vues et triggers : aperçu / Views and triggers overview
14. 14. Erreurs et pièges repérés dans les notes / Errors and traps in the notes
15. 15. Dictionnaire complet des commandes / Complete command dictionary
16. 16. Méthode de traduction d'un énoncé / Translating a prompt into SQL
17. 17. Exercices progressifs corrigés / Progressive corrected exercises
18. 18. Examen blanc / Mock exam
19. 19. Fiche ultra-courte de dernière minute / Last-minute cram sheet

1. Périmètre de l'examen et priorités / Scope and priorities

Les documents fournis couvrent deux blocs principaux : le LID, c'est-à-dire l'interrogation des données, et le LDD, c'est-à-dire la création et la modification physique des bases et des tables.

The supplied documents cover two main blocks: data querying (DQL) and database/table definition and alteration (DDL).

Ils contiennent aussi des exercices sur les fonctions, les regroupements, les sous-requêtes, les contraintes, les vues, les triggers et la modélisation MCD/MLD.

They also contain exercises on functions, grouping, subqueries, constraints, views, triggers, and conceptual/logical modelling.

Priorité	FR	EN	Poids conseillé
A	SELECT, WHERE, alias, DISTINCT, NULL, ORDER BY	SELECT, WHERE, aliases, DISTINCT, NULL, sorting	Très élevé
A	COUNT/SUM/AVG/MIN/MAX, GROUP BY, HAVING	Aggregates, grouping, group filtering	Très élevé
A	Sous-requêtes : IN, NOT IN, MAX/MIN, ALL	Subqueries: IN, NOT IN, MAX/MIN, ALL	Élevé
A	CREATE/ALTER, PK, FK, UNIQUE, CHECK, DEFAULT, AUTO_INCREMENT	Core DDL and constraints	Élevé
B	Fonctions texte, date, math, IFNULL	Text, date, numeric, NULL functions	Moyen à élevé
B	Jointures INNER et LEFT avec alias	INNER and LEFT joins with aliases	Minimum nécessaire
C	Vues et triggers	Views and triggers	Révision rapide
C	MCD, cardinalités, 1FN-3FN	Conceptual model, cardinalities, normal forms	Théorie de secours

Stratégie : n'apprenez pas des requêtes par cœur. Apprenez la séquence SELECT → FROM → WHERE → GROUP BY → HAVING → ORDER BY.

Strategy: do not memorize whole queries. Memorize the sequence SELECT → FROM → WHERE → GROUP BY → HAVING → ORDER BY.

2. Anatomie d'une requête SELECT / Anatomy of a SELECT query

Une requête LID est volatile : elle produit un résultat sans modifier les données. La partie projection indique ce qui est affiché ; la restriction indique quelles lignes sont conservées.

A DQL query is transient: it returns a result without changing data. Projection defines what is displayed; restriction defines which rows are kept.

```
SELECT [DISTINCT] expression1 [AS alias1], expression2 [AS alias2]
FROM table1 [AS t1]
[JOIN table2 AS t2 ON condition_de_liaison]
[WHERE conditions_sur_les_lignes]
[GROUP BY colonnes_de_regroupement]
[HAVING conditions_sur_les_groupes]
[ORDER BY expression ASC | DESC]
[LIMIT nombre];
```

2.1 Ordre d'écriture et ordre logique / Written order and logical order

Ordre écrit / Written	Ordre logique simplifié / Simplified logical order	Rôle / Role
SELECT	5	Choisir/calculer les colonnes affichées. / Select or calculate output columns.
FROM + JOIN	1	Construire la source. / Build the source rows.
WHERE	2	Filtrer les lignes. / Filter rows.
GROUP BY	3	Former les groupes. / Form groups.

Ordre écrit / Written	Ordre logique simplifié / Simplified logical order	Rôle / Role
HAVING	4	Filtrer les groupes. / Filter groups.
ORDER BY	6	Trier le résultat final. / Sort final result.
LIMIT	7	Limiter le nombre de lignes. / Limit row count.

Conséquence importante : un alias créé dans SELECT est utilisable dans ORDER BY, mais pas normalement dans WHERE, car WHERE est traité avant SELECT.

Important consequence: an alias created in SELECT can be used in ORDER BY, but normally not in WHERE, because WHERE is processed first.

```
SELECT nomm, prixm * 1.20 AS prix_ttc
FROM medicament
WHERE prixm * 1.20 > 100    -- pas WHERE prix_ttc > 100
ORDER BY prix_ttc DESC;
```

3. Alias de colonnes et de tables / Column and table aliases

Un alias est un nom temporaire valable pendant l'exécution de la requête. Il ne renomme pas réellement la colonne ou la table dans la base.

An alias is a temporary name valid during query execution. It does not permanently rename the database column or table.

3.1 Alias de colonne / Column alias

```
SELECT nomm AS medicament,
       prixm AS prix_ht,
       ROUND(prixm * 1.20, 2) AS prix_ttc
FROM medicament;
```

AS est recommandé pour la lisibilité. MySQL accepte aussi un espace sans AS, mais cette forme est moins explicite.

AS is recommended for clarity. MySQL also accepts a space without AS, but that form is less explicit.

```
SELECT SUM(prixc) AS chiffre_affaires
FROM consultation;
```

3.2 Alias contenant des espaces / Aliases containing spaces

Pour un intitulé avec espaces, utilisez de préférence les accents graves MySQL. Les notes utilisent parfois des apostrophes simples, tolérées dans certains contextes mais ambiguës.

For an alias containing spaces, preferably use MySQL backticks. The notes sometimes use single quotes, which may work in some contexts but are ambiguous.

```
SELECT nomm AS `nom du medicament`
FROM medicament;
```

3.3 Alias de table / Table alias

```
SELECT a.numa, a.noma, a.datenaissa
FROM animal AS a;
```

Le préfixe a. indique exactement de quelle table vient chaque colonne. Il devient indispensable quand plusieurs tables ont une colonne de même nom, par exemple numr.

The a. prefix states exactly which table a column comes from. It becomes essential when multiple tables share a column name, such as numr.

```
SELECT a.noma AS animal, r.nomr AS race
FROM animal AS a
INNER JOIN race AS r ON r.numr = a.numr;
```

3.4 Règles à mémoriser / Rules to memorize

Règle / Rule	Exemple / Example	Pourquoi / Why
Alias de colonne après l'expression	prixm * 1.2 AS prix_ttc	Nomme le résultat calculé. / Names a calculated result.
Alias de table dans FROM	FROM animal AS a	Raccourcit les références. / Shortens references.
Préfixer en multi-table	a.numr, r.numr	Évite les ambiguïtés. / Avoids ambiguity.
Alias dans ORDER BY	ORDER BY prix_ttc	ORDER BY vient après SELECT. / ORDER BY follows SELECT.
Pas d'alias SELECT dans WHERE	WHERE prixm*1.2 > 100	WHERE précède SELECT. / WHERE precedes SELECT.
Éviter les alias cryptiques	consultation AS c	Une lettre liée au nom de table. / Use a meaningful letter.

Forme réflexe pour l'examen : **SELECT t.colonne AS alias_colonne FROM table AS t;**

Exam reflex: *SELECT t.column AS column_alias FROM table AS t;*

4. Sélection, DISTINCT, NULL et tris / Selection, DISTINCT, NULL, and sorting

4.1 Projection simple / Simple projection

```
SELECT * FROM animal;  
SELECT noma, tatouage FROM animal;
```

L'étoile affiche toutes les colonnes. Dans une réponse d'examen, préférez les colonnes demandées, sauf si l'énoncé exige toutes les informations.

The asterisk displays all columns. In an exam answer, prefer the requested columns unless the prompt asks for all information.

4.2 DISTINCT

```
SELECT DISTINCT nomp FROM proprietaire;  
SELECT DISTINCT nomp, prenomp FROM proprietaire;
```

DISTINCT s'applique à la combinaison complète des colonnes du SELECT. Deux lignes sont doublons seulement si toutes les valeurs projetées sont identiques.

DISTINCT applies to the full combination of SELECT columns. Two rows are duplicates only when every projected value is identical.

4.3 COUNT et NULL

Expression	Compte / Counts	Ignore NULL ?	Exemple
COUNT(*)	Toutes les lignes / All rows	Non / No	COUNT(*) FROM animal
COUNT(colonne)	Valeurs non NULL / Non-NULL values	Oui / Yes	COUNT(tatouage)
COUNT(DISTINCT colonne)	Valeurs différentes non NULL / Distinct non-NULL values	Oui / Yes	COUNT(DISTINCT nomp)

```
SELECT COUNT(*) AS lignes,  
       COUNT(tatouage) AS tatouages_connus,  
       COUNT(DISTINCT tatouage) AS tatouages_différents  
FROM animal;
```

4.4 NULL

```
SELECT * FROM animal WHERE tatouage IS NULL;  
SELECT * FROM animal WHERE datenaissa IS NOT NULL;
```

Ne jamais écrire colonne = NULL ou colonne <> NULL. Utilisez IS NULL et IS NOT NULL.

Never write column = NULL or column <> NULL. Use IS NULL and IS NOT NULL.

4.5 ORDER BY et LIMIT

```
SELECT noma, datenaissa  
FROM animal  
WHERE datenaissa IS NOT NULL  
ORDER BY datenaissa ASC, noma ASC;  
  
SELECT * FROM consultation  
ORDER BY prixc DESC  
LIMIT 5;
```

ORDER BY 2 est possible, mais ORDER BY prixc est plus lisible et résiste mieux aux changements du SELECT.

ORDER BY 2 is possible, but ORDER BY prixc is clearer and safer if the SELECT list changes.

5. Filtres et opérateurs / Filters and operators

5.1 Comparaisons / Comparisons

Opérateur	FR	EN	Exemple
=	égal	equal	prixc = 100
<> ou !=	différent	not equal	nomp <> 'Fevrier'
>	supérieur	greater than	prixc > 200
>=	supérieur ou égal	greater than or equal	prixc >= 200
<	inférieur	less than	prixm < 50
<=	inférieur ou égal	less than or equal	annee <= 2012

5.2 AND, OR, NOT et priorité / AND, OR, NOT, and precedence

AND est évalué avant OR. Les parenthèses rendent l'intention explicite et évitent la majorité des erreurs.

AND is evaluated before OR. Parentheses make the intention explicit and prevent most errors.

```
SELECT *  
FROM proprietaire  
WHERE (nomp = 'Fevrier' OR nomp = 'Mars')  
AND prenomp <> 'Jean-Yves';
```

5.3 BETWEEN, IN et LIKE

```
SELECT * FROM consultation WHERE prixc BETWEEN 200 AND 300;  
SELECT * FROM medicament WHERE prixm IN (35, 86, 119);  
SELECT * FROM proprietaire WHERE villep NOT IN ('Paris','Lyon','Nantes');  
SELECT * FROM veterinaire WHERE nomv LIKE 'C%';
```

Motif LIKE	Signification / Meaning	Exemple
'C%'	Commence par C / Starts with C	Carre
'%e'	Se termine par e / Ends with e	Bipsie

Motif LIKE	Signification / Meaning	Exemple
'%a%'	Contient a / Contains a	Jafna
'_a%'	a en troisième position / a in third position	Blake
'_i%k'	Deuxième lettre i et fin k / second letter i and ends k	Niok
'% %'	Contient un espace / Contains a space	Jeune Alf

5.4 NOT IN et le piège NULL / NOT IN and the NULL trap

Si la sous-requête de NOT IN peut retourner NULL, le résultat peut devenir vide ou inattendu. Utilisez DISTINCT et excluez les NULL, ou préférez NOT EXISTS.

If a NOT IN subquery can return NULL, the result can become empty or unexpected. Exclude NULL values or prefer NOT EXISTS.

```
SELECT a.*
FROM animal AS a
WHERE NOT EXISTS (
  SELECT 1
  FROM consultation AS c
  WHERE c.numa = a.numa
);
```

6. Fonctions numériques, texte, dates et NULL / Numeric, text, date, and NULL functions

6.1 Calculs et fonctions numériques / Arithmetic and numeric functions

Fonction	FR	EN	Exemple
ROUND(x,n)	Arrondit à n décimales	Rounds to n decimals	ROUND(prixm*1.2,2)
TRUNCATE(x,n)	Coupe après n décimales	Truncates after n decimals	TRUNCATE(45.562,2)=45.56
MOD(a,b)	Reste de la division	Remainder	MOD(7,2)=1
POWER(a,b)	Puissance	Power	POWER(3,5)=243
SQRT(x)	Racine carrée	Square root	SQRT(25)=5

```
SELECT nomm,
  prixm AS prix_ht,
  ROUND(prixm * 0.20, 2) AS tva,
  ROUND(prixm * 1.20, 2) AS prix_ttc
FROM medicament;
```

6.2 Chaînes de caractères / String functions

Fonction	FR	EN	Exemple
UPPER(s)	Majuscules	Uppercase	UPPER(nomp)
LOWER(s)	Minuscules	Lowercase	LOWER(natures)
CONCAT(a,b,...)	Concatène	Concatenates	CONCAT(prenom, ' ', nomp)
LENGTH(s)	Longueur en octets	Byte length	LENGTH(noma)
CHAR_LENGTH(s)	Nombre de caractères	Character count	CHAR_LENGTH(noma)
SUBSTR(s,pos,n)	Extrait une sous-chaîne	Extracts substring	SUBSTR(noma,1,1)

Fonction	FR	EN	Exemple
LEFT(s,n)	n caractères à gauche	n leftmost characters	LEFT(datec,4)
RIGHT(s,n)	n caractères à droite	n rightmost characters	RIGHT(nom,4)
REPLACE(s,a,b)	Remplace a par b	Replaces a with b	REPLACE(region,' ','')

```
SELECT natures,
       CONCAT(UPPER(SUBSTR(natures,1,1)),
              LOWER(SUBSTR(natures,2))) AS nature_normalisee
FROM soin;
```

6.3 Dates / Dates

Fonction / Forme	Utilité / Purpose	Exemple
YEAR(date)	Extraire l'année / Extract year	YEAR(datec)=2024
MONTH(date)	Extraire le mois / Extract month	MONTH(datefac)=1
CURDATE()	Date du jour / Current date	CURDATE()
date BETWEEN '2024-01-01' AND '2024-12-31'	Intervalle explicite / Explicit range	Filtre annuel / Annual filter
date LIKE '2024-%'	Possible mais moins typé / Possible but less type-aware	À éviter si YEAR convient / Avoid when YEAR fits

```
SELECT YEAR(datec) AS annee, COUNT(*) AS nb_consultations
FROM consultation
GROUP BY YEAR(datec)
ORDER BY annee;
```

6.4 IFNULL et COALESCE

```
SELECT nom, sal, comm,
       sal + IFNULL(comm, 0) AS revenu_total
FROM emp;
```

Toute opération arithmétique avec NULL produit normalement NULL. IFNULL(valeur, remplacement) évite ce problème. COALESCE peut tester plusieurs valeurs successives.

Arithmetic involving NULL normally returns NULL. IFNULL(value, replacement) prevents this. COALESCE can test several fallback values.

7. Agrégats, GROUP BY, HAVING et GROUP_CONCAT

Une fonction d'agrégat résume plusieurs lignes en une valeur. Sans GROUP BY, toute la table forme un seul groupe. Avec GROUP BY, chaque combinaison distincte forme un groupe.

An aggregate function summarizes multiple rows into one value. Without GROUP BY, the entire table is one group. With GROUP BY, each distinct combination forms a group.

Fonction	FR	EN
COUNT	Nombre de lignes ou valeurs	Number of rows or values
SUM	Somme	Sum
AVG	Moyenne	Average
MIN	Minimum	Minimum
MAX	Maximum	Maximum

7.1 GROUP BY

```
SELECT numv,  
       COUNT(*) AS nb_consultations,  
       SUM(prix) AS chiffre_affaires  
FROM consultation  
GROUP BY numv  
ORDER BY chiffre_affaires DESC;
```

7.2 WHERE contre HAVING / WHERE versus HAVING

Clause	Filtre quoi ? / Filters what?	Moment / Timing	Exemple
WHERE	Lignes individuelles / Individual rows	Avant GROUP BY / Before grouping	WHERE prix >= 100
HAVING	Groupes calculés / Calculated groups	Après GROUP BY / After grouping	HAVING COUNT(*) > 5

```
SELECT nump, COUNT(*) AS nb_animaux  
FROM animal  
GROUP BY nump  
HAVING COUNT(*) > 3;
```

7.3 Colonnes non agrégées / Non-aggregated columns

Règle sûre : toute colonne du SELECT qui n'est pas dans une fonction d'agrégat doit apparaître dans GROUP BY.
Safe rule: every SELECT column not inside an aggregate function should appear in GROUP BY.

```
SELECT r.numr, r.nomr, COUNT(a.numa) AS nb_animaux  
FROM race AS r  
LEFT JOIN animal AS a ON a.numr = r.numr  
GROUP BY r.numr, r.nomr;
```

7.4 GROUP_CONCAT

```
SELECT nump,  
       GROUP_CONCAT(noma ORDER BY noma SEPARATOR ', ') AS liste_animaux  
FROM animal  
GROUP BY nump;
```

La syntaxe correcte MySQL place le séparateur après le mot SEPARATOR. Écrire GROUP_CONCAT(noma, ' ') concatène aussi un espace comme une expression par ligne, ce qui n'a pas le même sens.

Correct MySQL syntax places the separator after the SEPARATOR keyword. GROUP_CONCAT(noma, ' ') also concatenates a space expression for every row and does not mean the same thing.

7.5 WITH ROLLUP

```
SELECT IFNULL(job, 'TOTAL') AS job, COUNT(*) AS effectif, AVG(sal) AS salaire_moyen  
FROM emp  
GROUP BY job WITH ROLLUP;
```

WITH ROLLUP ajoute une ligne de total. C'est utile, mais moins prioritaire que GROUP BY et HAVING.

WITH ROLLUP adds a total row. It is useful, but lower priority than GROUP BY and HAVING.

8. Sous-requêtes / Subqueries

8.1 Sous-requête scalaire / Scalar subquery

```
SELECT *  
FROM medicament  
WHERE prixm = (SELECT MAX(prixm) FROM medicament);
```

La sous-requête renvoie une seule valeur. Les opérateurs =, <, >, <=, >= conviennent.

The subquery returns one value. Operators =, <, >, <=, and >= are appropriate.

8.2 Sous-requête liste / List subquery

```
SELECT *
FROM animal
WHERE numv IN (
  SELECT numv
  FROM propriétaire
  WHERE villev = 'Paris'
);
```

La sous-requête renvoie plusieurs valeurs. Utilisez IN ou NOT IN.

The subquery returns multiple values. Use IN or NOT IN.

8.3 ALL

```
SELECT numv, prix
FROM consultation
WHERE prix > ALL (
  SELECT prix
  FROM consultation
  WHERE numv = 1
);
```

$x > \text{ALL}(\text{liste})$ signifie x supérieur à chaque valeur, donc supérieur au maximum de la liste.

$x > \text{ALL}(\text{list})$ means x is greater than every value, therefore greater than the list maximum.

8.4 EXISTS et NOT EXISTS

```
SELECT a.*
FROM animal AS a
WHERE NOT EXISTS (
  SELECT 1
  FROM consultation AS c
  WHERE c.numa = a.numa
);
```

EXISTS teste l'existence d'au moins une ligne. SELECT 1 est conventionnel : le contenu projeté n'est pas utilisé.

EXISTS tests whether at least one row exists. SELECT 1 is conventional because the projected value is not used.

8.5 Maximum de groupes / Maximum among groups

```
SELECT numv, COUNT(*) AS nb_consultations
FROM consultation
GROUP BY numv
HAVING COUNT(*) >= ALL (
  SELECT COUNT(*)
  FROM consultation
  GROUP BY numv
);
```

Cette forme gère les ex aequo, contrairement à ORDER BY ... LIMIT 1 qui ne renvoie qu'une ligne.

This form handles ties, unlike ORDER BY ... LIMIT 1, which returns only one row.

9. Jointures minimales / Minimal joins

Le cours utilise parfois la syntaxe directe avec plusieurs tables dans FROM. Elle fonctionne, mais la syntaxe JOIN ... ON est plus claire et sépare la liaison des filtres.

The course sometimes uses the direct comma syntax in FROM. It works, but JOIN ... ON is clearer because it separates relationships from filters.

9.1 INNER JOIN : uniquement les correspondances / matching rows only

```
SELECT a.noma, r.nomr
FROM animal AS a
INNER JOIN race AS r ON r.numr = a.numr;
```

9.2 LEFT JOIN : conserver toute la table gauche / keep every left-row

```
SELECT r.numr, COUNT(a.noma) AS nb_animaux
FROM race AS r
LEFT JOIN animal AS a ON a.numr = r.numr
GROUP BY r.numr, r.nomr;
```

COUNT(a.noma) donne 0 pour une race sans animal. COUNT(*) donnerait 1 à cause de la ligne conservée par LEFT JOIN.

COUNT(a.noma) returns 0 for a breed with no animal. COUNT() would return 1 because LEFT JOIN preserves one row.*

9.3 Ancienne syntaxe directe / Old direct syntax

```
SELECT a.noma, r.nomr
FROM animal AS a, race AS r
WHERE r.numr = a.numr;
```

Pour l'examen : maîtrisez INNER JOIN, LEFT JOIN et la syntaxe directe. Inutile d'investir beaucoup de temps dans RIGHT JOIN ou les jointures complexes si elles ne sont pas demandées.

For the exam: master INNER JOIN, LEFT JOIN, and the direct syntax. Do not spend much time on RIGHT JOIN or complex joins unless required.

10. LDD/DDL : bases, tables, colonnes et contraintes

10.1 Base de données / Database

```
DROP DATABASE IF EXISTS ldd;
CREATE DATABASE ldd;
USE ldd;
```

10.2 CREATE TABLE

```
CREATE TABLE assistant (
  numas INT NOT NULL AUTO_INCREMENT,
  nomas VARCHAR(20) NOT NULL,
  mail VARCHAR(70) NOT NULL,
  numv INT NOT NULL,
  CONSTRAINT pk_assistant PRIMARY KEY (numas),
  CONSTRAINT uq_assistant_mail UNIQUE (mail),
  CONSTRAINT fk_assistant_veterinaire
    FOREIGN KEY (numv) REFERENCES veterinaire(numv)
);
```

10.3 Types essentiels / Essential types

Type	FR	EN	Exemple
INT	Entier	Integer	numa INT
DECIMAL(p,s)	Nombre exact avec décimales	Exact decimal	prix DECIMAL(8,2)
VARCHAR(n)	Texte variable	Variable-length text	nom VARCHAR(40)
CHAR(n)	Texte de longueur fixe	Fixed-length text	cp CHAR(5)
DATE	Date YYYY-MM-DD	Date	datec DATE

Type	FR	EN	Exemple
TIME	Heure	Time	heurec TIME
ENUM(...)	Une valeur parmi une liste	One value from a list	civilite ENUM('MADAME','MONSIEUR')

Un téléphone doit être VARCHAR ou CHAR, pas INT : il peut commencer par 0 et contenir +, espaces ou tirets.

A phone number should be VARCHAR or CHAR, not INT: it may start with 0 and contain +, spaces, or dashes.

10.4 Contraintes / Constraints

Contrainte	Effet / Effect	Forme / Form
NOT NULL	Valeur obligatoire / Required value	mail VARCHAR(70) NOT NULL
DEFAULT	Valeur par défaut / Default value	diplome VARCHAR(20) DEFAULT 'licence'
PRIMARY KEY	Identifie chaque ligne / Identifies each row	CONSTRAINT pk_x PRIMARY KEY(id)
FOREIGN KEY	Intégrité référentielle / Referential integrity	FOREIGN KEY(numv) REFERENCES veterinaire(numv)
UNIQUE	Interdit les doublons non NULL / Prevents duplicate non-NULL values	CONSTRAINT uq_mail UNIQUE(mail)
CHECK	Vérifie une condition / Checks a condition	CHECK(note BETWEEN 0 AND 20)
AUTO_INCREMENT	Génère une valeur numérique / Generates numeric values	id INT AUTO_INCREMENT

10.5 ALTER TABLE : ajouter, modifier, renommer, supprimer

```
ALTER TABLE assistant
ADD prenom VARCHAR(20) AFTER nom,
ADD mail VARCHAR(70) NOT NULL AFTER prenom;

ALTER TABLE assistant
MODIFY tel VARCHAR(20);

ALTER TABLE assistant
CHANGE nom nom_assistant VARCHAR(20);

ALTER TABLE assistant
RENAME stagiaire;

ALTER TABLE stagiaire
DROP COLUMN tel;
```

MODIFY change la définition sans renommer. CHANGE renomme et exige de répéter la définition complète du type et des options.

MODIFY changes the definition without renaming. CHANGE renames and requires the full type/options to be repeated.

10.6 Ajouter et supprimer des contraintes / Add and drop constraints

```
ALTER TABLE assistant
ADD CONSTRAINT pk_assistant PRIMARY KEY (numas);

ALTER TABLE assistant
ADD CONSTRAINT fk_assistant_veterinaire
FOREIGN KEY (numv) REFERENCES veterinaire(numv);

ALTER TABLE assistant
ADD CONSTRAINT uq_assistant_mail UNIQUE (mail);

ALTER TABLE eleve
ADD CONSTRAINT chk_note CHECK (noteg BETWEEN 0 AND 20);
```

À supprimer / To drop	Syntaxe MySQL sûre / Safe MySQL syntax
Clé étrangère / Foreign key	ALTER TABLE t DROP FOREIGN KEY fk_nom;
Clé primaire / Primary key	ALTER TABLE t DROP PRIMARY KEY;
UNIQUE	ALTER TABLE t DROP INDEX uq_nom;
CHECK	ALTER TABLE t DROP CHECK chk_nom;
Colonne / Column	ALTER TABLE t DROP COLUMN colonne;

Les notes écrivent parfois **DROP CONSTRAINT nom**. En MySQL, utilisez la forme spécifique ci-dessus. **DROP CONSTRAINT générique n'est pas la réponse la plus sûre.**

The notes sometimes use DROP CONSTRAINT name. In MySQL, use the constraint-specific form above. Generic DROP CONSTRAINT is not the safest answer.

10.7 Contrôles de clés étrangères / Foreign-key checks

```
SET FOREIGN_KEY_CHECKS = 0;
-- opérations exceptionnelles / exceptional operations
SET FOREIGN_KEY_CHECKS = 1;
```

Cette désactivation est limitée à la session sans le mot GLOBAL. Réactivez toujours les contrôles.

Without GLOBAL, this setting is session-scoped. Always re-enable the checks.

10.8 Inspection

```
SHOW TABLES;
DESC assistant;
SHOW CREATE TABLE assistant;
SHOW KEYS FROM assistant;
SELECT * FROM assistant\G
```

11. INSERT, UPDATE, DELETE et copies de tables

11.1 INSERT

```
INSERT INTO veterinaire (numv, nomv, prenomv)
VALUES (1, 'Dupond', 'Richard'),
       (2, 'Durand', 'Albert');

INSERT INTO eleve (nomel) VALUES ('Tutu');
```

Indiquer explicitement les colonnes rend l'insertion plus sûre. Pour AUTO_INCREMENT, omettez la colonne ou fournissez NULL.

Explicitly listing columns makes insertion safer. For AUTO_INCREMENT, omit the column or provide NULL.

11.2 UPDATE

```
UPDATE medicament  
SET prixm = prixm * 1.05  
WHERE numm = 3;
```

Sans WHERE, UPDATE modifie toutes les lignes. Faites d'abord un SELECT avec le même WHERE.

Without WHERE, UPDATE changes every row. First run a SELECT using the same WHERE.

11.3 DELETE

```
DELETE FROM soin  
WHERE numm = 10;
```

Sans WHERE, DELETE supprime toutes les lignes. DROP TABLE supprime la table elle-même.

Without WHERE, DELETE removes every row. DROP TABLE removes the table itself.

11.4 Copier une table / Copy a table

```
-- Structure seulement / Structure only  
CREATE TABLE stagiaire_copie LIKE stagiaire;  
  
-- Données ensuite / Then data  
INSERT INTO stagiaire_copie  
SELECT * FROM stagiaire;  
  
-- Copie rapide des données, contraintes souvent non conservées  
-- Quick data copy; constraints are often not preserved  
CREATE TABLE animhisto AS  
SELECT * FROM animal;
```

12. Modélisation relationnelle : minimum théorique / Relational modelling essentials

Le MCD décrit le métier avec des entités, propriétés, identifiants, associations et cardinalités. Le MLD transforme ce modèle en relations, clés primaires et clés étrangères. Le modèle physique devient le script SQL.

The conceptual model describes the business using entities, properties, identifiers, relationships, and cardinalities. The logical model turns it into relations, primary keys, and foreign keys. The physical model becomes SQL.

Terme FR	English	Idée à retenir / Key idea
Entité	Entity	Type d'objet du système : ANIMAL, PROPRIETAIRE.
Propriété	Attribute	Information atomique : noma, datenaissa.
Identifiant	Identifier	Valeur stable et unique ; devient souvent la PK.
Association	Relationship	Lien métier entre entités.
Cardinalité	Cardinality	Minimum et maximum de participation : 0,1 ; 1,1 ; 0,n ; 1,n.
Clé primaire	Primary key	Identifie un tuple de façon unique.
Clé étrangère	Foreign key	Référence la PK d'une autre table.
Domaine	Domain	Ensemble des valeurs autorisées pour une colonne.

12.1 Formes normales / Normal forms

1FN : chaque cellule contient une valeur atomique. 2FN : un attribut non-clé dépend de toute la clé composée. 3FN : les attributs non-clés ne dépendent pas d'autres attributs non-clés.

1NF: every cell contains one atomic value. 2NF: a non-key attribute depends on the whole composite key. 3NF: non-key attributes do not depend on other non-key attributes.

Pour une épreuve SQL pratique, reprenez surtout : une table = un sujet, une clé primaire stable, et les relations représentées par des clés étrangères.

For a practical SQL exam, mainly remember: one table per subject, a stable primary key, and relationships represented by foreign keys.

13. Vues et triggers : aperçu / Views and triggers overview

13.1 Vue / View

```
CREATE OR REPLACE VIEW activites_modiques AS
SELECT nomStation AS station, libelle AS activite
FROM activite
WHERE prix < 140
WITH CHECK OPTION;
```

Une vue est une requête enregistrée. WITH CHECK OPTION impose que les lignes insérées ou modifiées à travers la vue restent visibles dans cette vue.

A view is a saved query. WITH CHECK OPTION requires rows inserted or updated through the view to remain visible in it.

13.2 Trigger / Trigger

```
DELIMITER //
CREATE TRIGGER trg_exemple
AFTER INSERT ON consultation
FOR EACH ROW
BEGIN
    -- actions utilisant NEW.colonne / actions using NEW.column
END//
DELIMITER ;
```

Un trigger se déclenche automatiquement BEFORE ou AFTER INSERT, UPDATE ou DELETE. NEW représente la nouvelle ligne ; OLD représente l'ancienne ligne.

A trigger automatically fires BEFORE or AFTER INSERT, UPDATE, or DELETE. NEW represents the new row; OLD represents the old row.

Priorité faible : sachez reconnaître la structure, mais ne sacrifiez pas GROUP BY, HAVING, les alias ou ALTER TABLE pour apprendre des triggers complexes.

Low priority: recognize the structure, but do not sacrifice GROUP BY, HAVING, aliases, or ALTER TABLE to learn complex triggers.

14. Erreurs et pièges repérés dans les notes / Errors and traps found in the notes

Note brute / Raw note	Risque / Risk	Forme sûre / Safe form
COUNT()	Incorrect ou non portable. / Incorrect or non-portable.	COUNT(*) ou COUNT(colonne).
GROUP_CONCAT(immat, ' ')	Le second argument est concaténé à chaque ligne. / The second expression is appended per row.	GROUP_CONCAT(immat SEPARATOR ' ')
GROUP BY marque, modele avec GROUP_CONCAT(modele)	Regroupe déjà chaque modèle ; concaténation inutile. / Already groups each model.	GROUP BY marque seulement.
Condition annoncée mais absente	Exemple voitures noires ou blanches : la couleur doit figurer dans WHERE. / Color must appear in WHERE.	Ajouter AND couleur IN ('noir','blanc').
DROP CONSTRAINT nom	Pas la forme MySQL la plus sûre. / Not safest MySQL form.	DROP FOREIGN KEY, DROP INDEX, DROP CHECK...
ENUM et SET présentés comme équivalents	SET peut stocker plusieurs choix simultanément. /	Pour civilité, préférer ENUM.

Note brute / Raw note	Risque / Risk	Forme sûre / Safe form
	SET may hold multiple choices.	
INT(10) pour téléphone	INT(10) n'impose pas 10 chiffres et perd le zéro initial. / Does not enforce 10 digits and loses leading zero.	VARCHAR(20) ou CHAR.
LENGTH pour texte accentué	Compte les octets en UTF-8. / Counts bytes in UTF-8.	CHAR_LENGTH pour les caractères.
NOT IN avec NULL	Peut donner un résultat inattendu. / Can return unexpected results.	NOT EXISTS ou exclure NULL.
ORDER BY 2	Valide mais fragile. / Valid but fragile.	ORDER BY alias ou colonne.
Alias entre apostrophes simples	Toléré dans certains SELECT, mais ambigu. / Sometimes accepted but ambiguous.	Alias simple ou `alias avec espaces`.
SELECT colonne non agrégée hors GROUP BY	Peut échouer avec ONLY_FULL_GROUP_BY. / May fail.	Ajouter la colonne au GROUP BY.

15. Dictionnaire complet des commandes / Complete command dictionary

Commande / Command	Français	English	Exemple / Example
SELECT	Choisit les colonnes et expressions à afficher.	Chooses output columns and expressions.	SELECT noma FROM animal;
FROM	Indique la ou les sources.	Specifies source table(s).	FROM animal AS a
*	Toutes les colonnes.	All columns.	SELECT * FROM animal;
AS	Crée un alias temporaire.	Creates a temporary alias.	SUM(prixc) AS ca
DISTINCT	Élimine les doublons de la projection.	Removes duplicate projected rows.	SELECT DISTINCT villep ...
WHERE	Filtre les lignes avant regroupement.	Filters rows before grouping.	WHERE prixc > 100
AND / OR / NOT	Combine ou nie des conditions.	Combines or negates conditions.	(A OR B) AND NOT C
=, <>, !=, <, <=, >, >=	Comparaisons.	Comparisons.	prixc >= 50
BETWEEN	Intervalle inclusif.	Inclusive range.	prixc BETWEEN 200 AND 300
IN / NOT IN	Compare à une liste ou sous-requête.	Compares with a list or subquery.	villep IN ('Paris','Lyon')
LIKE / NOT LIKE	Recherche par motif.	Pattern matching.	nomv LIKE 'C%'
IS NULL / IS NOT NULL	Teste NULL.	Tests NULL.	email IS NULL
ORDER BY	Trie le résultat.	Sorts result.	ORDER BY prixc DESC
ASC / DESC	Croissant / décroissant.	Ascending / descending.	ORDER BY nom ASC
LIMIT	Limite le nombre de lignes.	Limits row count.	LIMIT 5
COUNT(*)	Compte les lignes.	Counts rows.	COUNT(*)
COUNT(col)	Compte les valeurs non NULL.	Counts non-NULL values.	COUNT(tatouage)
COUNT(DISTINCT col)	Compte les valeurs différentes non NULL.	Counts distinct non-NULL values.	COUNT(DISTINCT nomp)
SUM	Somme.	Sum.	SUM(prixc)
AVG	Moyenne.	Average.	AVG(sal)
MIN / MAX	Minimum / maximum.	Minimum / maximum.	MAX(prixc)
GROUP BY	Crée les groupes.	Creates groups.	GROUP BY numv
HAVING	Filtre les groupes.	Filters groups.	HAVING COUNT(*) > 5
GROUP_CONCAT	Concatène les valeurs d'un groupe.	Concatenates group values.	GROUP_CONCAT(noma SEPARATOR ' , ')

Commande / Command	Français	English	Exemple / Example
WITH ROLLUP	Ajoute des sous-totaux/totaux.	Adds subtotal/total rows.	GROUP BY job WITH ROLLUP
UPPER / LOWER	Change la casse.	Changes case.	UPPER(nomp)
CONCAT	Assemble des chaînes.	Joins strings.	CONCAT(prenomp, ' ', nomp)
LENGTH / CHAR_LENGTH	Octets / caractères.	Bytes / characters.	CHAR_LENGTH(noma)
SUBSTR / LEFT / RIGHT	Extrait une partie de chaîne.	Extracts part of a string.	SUBSTR(noma,1,1)
REPLACE	Remplace une sous-chaîne.	Replaces substring.	REPLACE(region, ' ', ',')
ROUND / TRUNCATE	Arrondit / tronque.	Rounds / truncates.	ROUND(prixm,2)
MOD / POWER / SQRT	Fonctions mathématiques.	Math functions.	MOD(7,2)
YEAR / MONTH / CURDATE	Fonctions de date.	Date functions.	YEAR(datec)
IFNULL / COALESCE	Remplace un NULL.	Provides NULL fallback.	IFNULL(comm,0)
Sous-requête / Subquery	Requête imbriquée.	Nested query.	WHERE prixm=(SELECT MAX...)
ALL	Compare à toutes les valeurs.	Compares with every value.	prixc > ALL (...)
EXISTS / NOT EXISTS	Teste l'existence.	Tests existence.	WHERE NOT EXISTS (...)
INNER JOIN	Conserve les correspondances.	Keeps matching rows.	JOIN race r ON ...
LEFT JOIN	Conserve toutes les lignes à gauche.	Keeps all left rows.	race r LEFT JOIN animal a ...
CREATE DATABASE	Crée une base.	Creates a database.	CREATE DATABASE ldd;
DROP DATABASE IF EXISTS	Supprime une base si elle existe.	Drops database if present.	DROP DATABASE IF EXISTS ldd;
USE	Sélectionne la base active.	Selects active database.	USE ldd;
CREATE TABLE	Crée une table.	Creates a table.	CREATE TABLE assistant (...);
DROP TABLE	Supprime table et structure.	Drops table and structure.	DROP TABLE test;
ALTER TABLE ADD	Ajoute colonne ou contrainte.	Adds column or constraint.	ALTER TABLE t ADD col INT;
ALTER TABLE MODIFY	Modifie la définition d'une colonne.	Changes column definition.	MODIFY tel VARCHAR(20)
ALTER TABLE CHANGE	Renomme et redéfinit une colonne.	Renames and redefines a column.	CHANGE ancien nouveau VARCHAR(20)
ALTER TABLE RENAME	Renomme une table.	Renames a table.	ALTER TABLE a RENAME b;
DROP COLUMN	Supprime une colonne.	Drops a column.	ALTER TABLE t DROP COLUMN x;
PRIMARY KEY	Clé principale unique et non NULL.	Unique non-NULL primary key.	PRIMARY KEY(id)
FOREIGN KEY ... REFERENCES	Crée une référence.	Creates a reference.	FOREIGN KEY(numv) REFERENCES vétérinaire(numv)
UNIQUE	Interdit les doublons.	Prevents duplicates.	UNIQUE(mail)
CHECK	Contrôle une condition.	Checks a condition.	CHECK(note BETWEEN 0 AND 20)
DEFAULT	Valeur par défaut.	Default value.	DEFAULT 'licence'
AUTO_INCREMENT	Génère les identifiants.	Generates identifiers.	id INT AUTO_INCREMENT
INSERT INTO	Ajoute des lignes.	Inserts rows.	INSERT INTO t(col) VALUES(...);
UPDATE	Modifie des lignes.	Updates rows.	UPDATE t SET x=... WHERE ...;
DELETE FROM	Supprime des lignes.	Deletes rows.	DELETE FROM t WHERE ...;
CREATE TABLE ... LIKE	Copie la structure.	Copies structure.	CREATE TABLE b LIKE a;
CREATE TABLE ... AS SELECT	Crée depuis un résultat.	Creates from query result.	CREATE TABLE b AS SELECT * FROM a;
SHOW TABLES	Liste les tables.	Lists tables.	SHOW TABLES;
DESC / DESCRIBE	Affiche la structure.	Shows table structure.	DESC animal;
SHOW CREATE TABLE	Affiche le SQL de création.	Shows creation SQL.	SHOW CREATE TABLE animal;
SHOW KEYS	Affiche index et clés.	Shows indexes and keys.	SHOW KEYS FROM animal;
SET FOREIGN_KEY_CHECKS	Active/désactive contrôles FK.	Enables/disables FK checks.	SET FOREIGN_KEY_CHECKS=0;

Commande / Command	Français	English	Exemple / Example
IG	Affichage vertical dans le client MySQL.	Vertical output in MySQL client.	SELECT * FROM animal\G
\T fichier	Journalise la session client.	Logs client session.	\T c:\temp\cours.sql
CREATE VIEW	Crée une vue.	Creates a view.	CREATE VIEW v AS SELECT ...;
WITH CHECK OPTION	Contrôle les écritures via une vue.	Checks writes through a view.	... WITH CHECK OPTION
CREATE TRIGGER	Crée un déclencheur.	Creates a trigger.	CREATE TRIGGER ...
DELIMITER	Change temporairement le délimiteur client.	Temporarily changes client delimiter.	DELIMITER //

16. Méthode de traduction d'un énoncé / Translating a prompt into SQL

Avant d'écrire, découpez l'énoncé en six questions. Cette méthode évite d'oublier un filtre, un regroupement ou un tri.

Before writing SQL, split the prompt into six questions. This prevents missing a filter, grouping, or sort.

Question mentale / Mental question	Clause	Exemple
Qu'est-ce que je dois afficher ? / What must I display?	SELECT	nomr, COUNT(a.numa) AS nb
Où sont les données ? / Where is the data?	FROM / JOIN	race r LEFT JOIN animal a
Quelles lignes garder ? / Which rows?	WHERE	datec >= '2024-01-01'
Dois-je faire un calcul par catégorie ? / Per category?	GROUP BY	GROUP BY r.numr, r.nomr
Dois-je filtrer un calcul ? / Filter an aggregate?	HAVING	HAVING COUNT(a.numa) > 3
Dans quel ordre ? / What order?	ORDER BY	ORDER BY nb DESC

16.1 Exemple complet / Full example

Énoncé : afficher les vétérinaires ayant réalisé plus de cinq consultations, avec leur nombre de consultations et leur chiffre d'affaires, du chiffre d'affaires le plus élevé au plus faible.

Prompt: display veterinarians who performed more than five consultations, with their consultation count and revenue, from highest to lowest revenue.

```
SELECT v.numv,
       CONCAT(v.prenomv, ' ', v.nomv) AS veterinaire,
       COUNT(c.numc) AS nb_consultations,
       SUM(c.prixc) AS chiffre_affaires
FROM veterinaire AS v
INNER JOIN consultation AS c ON c.numv = v.numv
GROUP BY v.numv, v.prenomv, v.nomv
HAVING COUNT(c.numc) > 5
ORDER BY chiffre_affaires DESC;
```

16.2 Procédure de débogage / Debugging procedure

20. 1. Exécuter SELECT ... FROM sans WHERE.
 1. Run SELECT ... FROM without WHERE.
21. 2. Ajouter les filtres un par un.
 2. Add filters one at a time.
22. 3. Vérifier les NULL avec IS NULL/IS NOT NULL.
 3. Check NULLs with IS NULL/IS NOT NULL.

23. 4. Ajouter GROUP BY et les agrégats.
4. Add GROUP BY and aggregates.
24. 5. Ajouter HAVING seulement pour les agrégats.
5. Add HAVING only for aggregate filters.
25. 6. Ajouter ORDER BY à la fin.
6. Add ORDER BY last.
26. 7. Pour UPDATE/DELETE, tester d'abord le même WHERE avec SELECT.
7. For UPDATE/DELETE, first test the same WHERE using SELECT.

17. Exercices progressifs corrigés / Progressive corrected exercises

Utilisez `Base_pratique_revision_mysql.sql`. Cachez les corrections pendant le premier passage.

Use `Base_pratique_revision_mysql.sql`. Hide the solutions on your first attempt.

17.1 Drill alias - 12 questions / Alias drill - 12 questions

1. Afficher nomm sous l'alias médicament.

Display nomm as médicament.

```
SELECT nomm AS médicament FROM médicament;
```

2. Afficher prixm comme prix_ht et prixm*1.2 comme prix_ttc.

*Display prixm as prix_ht and prixm*1.2 as prix_ttc.*

```
SELECT prixm AS prix_ht, ROUND(prixm*1.2,2) AS prix_ttc FROM médicament;
```

3. Donner l'alias a à animal et afficher a.numa, a.noma.

Alias animal as a and display a.numa, a.noma.

```
SELECT a.numa, a.noma FROM animal AS a;
```

4. Afficher le nom complet du propriétaire sous nom_complet.

Display the owner full name as nom_complet.

```
SELECT CONCAT(prenom,' ',nom) AS nom_complet FROM propriétaire;
```

5. Afficher YEAR(datec) sous annee_consultation.

Display YEAR(datec) as annee_consultation.

```
SELECT YEAR(datec) AS annee_consultation FROM consultation;
```

6. Afficher COUNT(*) sous nb_animaux.

Display COUNT() as nb_animaux.*

```
SELECT COUNT(*) AS nb_animaux FROM animal;
```

7. Afficher SUM(prixc) sous chiffre_affaires.

Display SUM(prixc) as chiffre_affaires.

```
SELECT SUM(prixc) AS chiffre_affaires FROM consultation;
```

8. Avec c comme alias de consultation, afficher c.numc et c.prixc.

Alias consultation as c and display c.numc and c.prixc.

```
SELECT c.numc, c.prixc FROM consultation AS c;
```

9. Avec a et r, afficher a.noma AS animal et r.nomr AS race.

Using a and r, display a.noma AS animal and r.nomr AS race.

```
SELECT a.noma AS animal, r.nomr AS race FROM animal AS a JOIN race AS r ON r.numr=a.numr;
```

10. Trier un prix TTC calculé en utilisant son alias.

Sort a calculated tax-inclusive price using its alias.

```
SELECT nomm, prixm*1.2 AS prix_ttc FROM médicament ORDER BY prix_ttc DESC;
```

11. Filtrer un prix TTC sans utiliser son alias dans WHERE.

Filter a tax-inclusive price without using its alias in WHERE.

```
SELECT nomm, prixm*1.2 AS prix_ttc FROM medicament WHERE prixm*1.2>100;
```

12. Créer un alias avec espaces de façon sûre.

Create a safe alias containing spaces.

```
SELECT nomm AS `nom du medicament` FROM medicament;
```

17.2 Requêtes mixtes - 18 questions / Mixed queries - 18 questions

1. Afficher les animaux non tatoués.

Display animals without tattoos.

```
SELECT * FROM animal WHERE tatouage IS NULL;
```

2. Afficher les noms de propriétaires différents.

Display distinct owner surnames.

```
SELECT DISTINCT nomp FROM proprietaire;
```

3. Compter tous les animaux, les tatouages connus et les tatouages différents.

Count all animals, known tattoos, and distinct tattoos.

```
SELECT COUNT(*) AS lignes, COUNT(tatouage) AS connus, COUNT(DISTINCT tatouage) AS differents FROM animal;
```

4. Afficher les médicaments entre 40 et 100 euros.

Display medicines priced between 40 and 100.

```
SELECT * FROM medicament WHERE prixm BETWEEN 40 AND 100;
```

5. Afficher les propriétaires de Paris ou Lyon qui ne s'appellent pas Fevrier.

Display Paris or Lyon owners not named Fevrier.

```
SELECT * FROM proprietaire WHERE (villep IN ('Paris','Lyon')) AND nomp <> 'Fevrier';
```

6. Afficher les animaux dont le nom commence par B.

Display animal names starting with B.

```
SELECT * FROM animal WHERE noma LIKE 'B%';
```

7. Afficher les médicaments en majuscules.

Display medicine names in uppercase.

```
SELECT UPPER(nomm) AS nom_majuscule FROM medicament;
```

8. Afficher les initiales des propriétaires.

Display owner initials.

```
SELECT CONCAT(UPPER(LEFT(nomp,1)), UPPER(LEFT(prenom,1))) AS initiales FROM proprietaire;
```

9. Afficher les années et nombres de consultations.

Display consultation counts by year.

```
SELECT YEAR(datec) AS annee, COUNT(*) AS nb FROM consultation GROUP BY YEAR(datec);
```

10. Afficher les propriétaires ayant plus de trois animaux.

Display owners with more than three animals.

```
SELECT numr, COUNT(*) AS nb FROM animal GROUP BY numr HAVING COUNT(*)>3;
```

11. Afficher le médicament le plus cher.

Display the most expensive medicine.

```
SELECT * FROM medicament WHERE prixm=(SELECT MAX(prixm) FROM medicament);
```

12. Afficher les animaux sans consultation avec NOT EXISTS.

Display animals without consultations using NOT EXISTS.

```
SELECT a.* FROM animal AS a WHERE NOT EXISTS (SELECT 1 FROM consultation AS c WHERE c.numa=a.numa);
```

13. Afficher toutes les races et leur nombre d'animaux.

Display all breeds and their animal count.

```
SELECT r.numr,r.nomr,COUNT(a.numa) AS nb FROM race AS r LEFT JOIN animal AS a ON a.numr=r.numr GROUP BY r.numr,r.nomr;
```

14. Afficher le chiffre d'affaires par vétérinaire.

Display revenue per veterinarian.

```
SELECT numv,SUM(prix) AS ca FROM consultation GROUP BY numv ORDER BY ca DESC;
```

15. Afficher les années ayant au moins cinq consultations.

Display years with at least five consultations.

```
SELECT YEAR(datec) AS annee,COUNT(*) AS nb FROM consultation GROUP BY YEAR(datec) HAVING COUNT(*)>=5;
```

16. Afficher les animaux des propriétaires parisiens sans jointure.

Display animals owned by Paris owners without a join.

```
SELECT * FROM animal WHERE nump IN (SELECT nump FROM proprietaire WHERE villep='Paris');
```

17. Afficher les consultations plus chères que toutes celles du vétérinaire 1.

Display consultations more expensive than all those by veterinarian 1.

```
SELECT * FROM consultation WHERE prix > ALL (SELECT prix FROM consultation WHERE numv=1);
```

18. Lister les noms d'animaux par propriétaire dans une cellule.

List animal names per owner in one cell.

```
SELECT nump,GROUP_CONCAT(noma ORDER BY noma SEPARATOR ',') AS animaux FROM animal GROUP BY nump;
```

17.3 Exercices LDD - 10 questions / 10 DDL exercises

1. Créer une base bac_sable et la sélectionner.

Create and select a bac_sable database.

```
DROP DATABASE IF EXISTS bac_sable;
CREATE DATABASE bac_sable;
USE bac_sable;
```

2. Créer une table personne : id auto-incrémenté, nom obligatoire, email unique.

Create person table: auto-increment id, required name, unique email.

```
CREATE TABLE personne(id INT NOT NULL AUTO_INCREMENT, nom VARCHAR(50) NOT NULL, email VARCHAR(100), CONSTRAINT
pk_personne PRIMARY KEY(id), CONSTRAINT uq_personne_email UNIQUE(email));
```

3. Ajouter prenomas après nomas dans assistant.

Add prenomas after nomas in assistant.

```
ALTER TABLE assistant ADD prenomas VARCHAR(20) AFTER nomas;
```

4. Ajouter mail obligatoire et unique.

Add required unique mail.

```
ALTER TABLE assistant ADD mail VARCHAR(70) NOT NULL;
ALTER TABLE assistant ADD CONSTRAINT uq_assistant_mail UNIQUE(mail);
```

5. Modifier tel en VARCHAR(20).

Change tel to VARCHAR(20).

```
ALTER TABLE assistant MODIFY tel VARCHAR(20);
```

6. Renommer nomas en nom_assistant.

Rename nomas to nom_assistant.

```
ALTER TABLE assistant CHANGE nomas nom_assistant VARCHAR(20);
```

7. Ajouter numv et sa clé étrangère.

Add numv and its foreign key.

```
ALTER TABLE assistant ADD numv INT NOT NULL;
ALTER TABLE assistant ADD CONSTRAINT fk_assistant_veto FOREIGN KEY(numv) REFERENCES veterinaire(numv);
```

8. Supprimer cette clé étrangère.

Drop that foreign key.

```
ALTER TABLE assistant DROP FOREIGN KEY fk_assistant_veto;
```

9. Copier la structure de assistant.

Copy assistant structure.

```
CREATE TABLE assistant_copie LIKE assistant;
```

10. Copier ensuite les données.

Then copy the data.

```
INSERT INTO assistant_copie SELECT * FROM assistant;
```

17.4 Atelier de syntaxe correcte - réparer les requêtes / Correct syntax workshop - repair the queries

Méthode : pour chaque requête, repérez l'erreur, réécrivez toute la requête proprement, puis comparez avec la correction. Ne corrigez pas seulement le mot fautif : reconstruisez la forme complète.

Method: for each query, identify the error, rewrite the whole query cleanly, then compare it with the solution. Do not fix only the faulty word: rebuild the complete statement.

Barème personnel : 1 point pour la syntaxe exécutable, 1 point pour la logique correcte, 1 point pour la lisibilité (alias, indentation, noms explicites). Objectif : 48/60 ou plus.

Self-marking: 1 point for executable syntax, 1 point for correct logic, and 1 point for readability (aliases, indentation, meaningful names). Target: 48/60 or higher.

A. Requêtes à réparer - sans regarder la correction / Queries to repair - do not check solutions yet

1. Compter toutes les lignes de animal.

Count every row in animal.

```
-- Requête incorrecte / Incorrect query
SELECT COUNT() FROM animal;
```

2. Afficher les propriétaires sans email.

Display owners without an email.

```
-- Requête incorrecte / Incorrect query
SELECT * FROM proprietaire WHERE email = NULL;
```

3. Compter les animaux par race, puis trier du plus grand au plus petit.

Count animals by breed, then sort highest to lowest.

```
-- Requête incorrecte / Incorrect query
SELECT numr, COUNT(*) AS nb FROM animal ORDER BY nb DESC GROUP BY numr;
```

4. Afficher uniquement les médicaments dont le prix TTC dépasse 100.

Display only medicines whose tax-inclusive price exceeds 100.

```
-- Requête incorrecte / Incorrect query
SELECT nomm, prixm * 1.20 AS prix_ttc FROM medicament WHERE prix_ttc > 100;
```

5. Concaténer les noms d'animaux avec une virgule par propriétaire.

Concatenate animal names with commas for each owner.

```
-- Requête incorrecte / Incorrect query
SELECT nump, GROUP_CONCAT(noma, ', ') AS animaux FROM animal GROUP BY nump;
```

6. Afficher les propriétaires de Paris ou Lyon dont le nom n'est pas Fevrier.

Display Paris or Lyon owners whose surname is not Fevrier.

```
-- Requête incorrecte / Incorrect query
SELECT * FROM proprietaire WHERE villep = 'Paris' OR villep = 'Lyon' AND nomp <> 'Fevrier';
```

7. Afficher les races ayant au moins deux animaux.

Display breeds with at least two animals.

```
-- Requête incorrecte / Incorrect query
SELECT numr, COUNT(*) AS nb FROM animal HAVING COUNT(*) >= 2 GROUP BY numr;
```

8. Afficher le nom de la race et son nombre d'animaux.

Display each breed name and its animal count.

```
-- Requête incorrecte / Incorrect query
SELECT nomr, COUNT(*) AS nb FROM race;
```

9. Donner l'alias a à animal et afficher le nom.

Alias animal as a and display the name.

```
-- Requête incorrecte / Incorrect query  
SELECT a.noma FROM animal a AS;
```

10. Afficher les animaux avec leur race.

Display animals with their breed.

```
-- Requête incorrecte / Incorrect query  
SELECT a.noma, r.nomr FROM animal AS a INNER JOIN race AS r;
```

11. Afficher les animaux dont le propriétaire habite Paris, avec une sous-requête.

Display animals whose owner lives in Paris using a subquery.

```
-- Requête incorrecte / Incorrect query  
SELECT * FROM animal WHERE numr IN SELECT numr FROM propriétaire WHERE villeg = 'Paris';
```

12. Afficher les consultations comprises entre 200 et 300 euros.

Display consultations priced between 200 and 300 euros.

```
-- Requête incorrecte / Incorrect query  
SELECT * FROM consultation WHERE prix BETWEEN 200 TO 300;
```

13. Ajouter une colonne mail à assistant.

Add a mail column to assistant.

```
-- Requête incorrecte / Incorrect query  
ALTER assistant ADD mail VARCHAR(70);
```

14. Renommer nomas en nom_assistant.

Rename nomas to nom_assistant.

```
-- Requête incorrecte / Incorrect query  
ALTER TABLE assistant CHANGE nomas nom_assistant;
```

15. Supprimer la clé étrangère fk_assistant_veto.

Drop the fk_assistant_veto foreign key.

```
-- Requête incorrecte / Incorrect query  
ALTER TABLE assistant DROP CONSTRAINT fk_assistant_veto;
```

16. Insérer un assistant avec numas, nomas et tel.

Insert an assistant using numas, nomas, and tel.

```
-- Requête incorrecte / Incorrect query  
INSERT INTO assistant (numas, nomas, tel) VALUES (1, 'PEMBA');
```

17. Modifier le téléphone de l'assistant 1.

Update the telephone number of assistant 1.

```
-- Requête incorrecte / Incorrect query  
UPDATE assistant tel = '0606060606' WHERE numas = 1;
```

18. Supprimer l'assistant 1.

Delete assistant 1.

```
-- Requête incorrecte / Incorrect query  
DELETE * FROM assistant WHERE numas = 1;
```

19. Créer une table eleve avec une clé primaire auto-incrémentée.

Create an eleve table with an auto-increment primary key.

```
-- Requête incorrecte / Incorrect query  
CREATE TABLE eleve (idel INT AUTO_INCREMENT nomel VARCHAR(30), PRIMARY KEY idel);
```

20. Afficher toutes les races, même sans animal, avec leur nombre.

Display all breeds, including those with no animals, with their count.

```
-- Requête incorrecte / Incorrect query  
SELECT r.nomr, COUNT(*) AS nb FROM race AS r LEFT JOIN animal AS a ON a.numr = r.numr GROUP BY r.nomr;
```

B. Corrections commentées / Commented solutions

1.

```
SELECT COUNT(*) AS nb_animaux  
FROM animal;
```

*COUNT exige * ou une expression. / COUNT requires * or an expression.*

2.

```
SELECT *  
FROM proprietaire  
WHERE email IS NULL;
```

NULL se teste avec IS NULL. / Test NULL with IS NULL.

3.

```
SELECT numr, COUNT(*) AS nb  
FROM animal  
GROUP BY numr  
ORDER BY nb DESC;
```

GROUP BY précède ORDER BY. / GROUP BY comes before ORDER BY.

4.

```
SELECT nomm, prixm * 1.20 AS prix_ttc  
FROM medicament  
WHERE prixm * 1.20 > 100;
```

Un alias du SELECT ne s'emploie normalement pas dans WHERE. / A SELECT alias is normally unavailable in WHERE.

5.

```
SELECT nump,  
       GROUP_CONCAT(noma ORDER BY noma SEPARATOR ', ') AS animaux  
FROM animal  
GROUP BY nump;
```

Le séparateur suit le mot-clé SEPARATOR. / The separator follows the SEPARATOR keyword.

6.

```
SELECT *  
FROM proprietaire  
WHERE villep IN ('Paris', 'Lyon')  
AND nomp <> 'Fevrier';
```

Regroupez le OR, ou remplacez-le par IN. / Group the OR, or replace it with IN.

7.

```
SELECT numr, COUNT(*) AS nb  
FROM animal  
GROUP BY numr  
HAVING COUNT(*) >= 2;
```

HAVING vient après GROUP BY. / HAVING follows GROUP BY.

8.

```
SELECT r.numr, COUNT(a.numa) AS nb_animaux  
FROM race AS r  
LEFT JOIN animal AS a ON a.numr = r.numr  
GROUP BY r.numr, r.nomm;
```

Le nom doit être groupé ; LEFT JOIN permet aussi les races vides. / The name must be grouped; LEFT JOIN also keeps empty breeds.

9.

```
SELECT a.noma  
FROM animal AS a;
```

L'alias vient après la table : table AS alias. / Put the alias after the table: table AS alias.

10.

```
SELECT a.noma, r.nomr
FROM animal AS a
INNER JOIN race AS r ON r.numr = a.numr;
```

Une jointure explicite exige une condition ON. / An explicit join requires an ON condition.

11.

```
SELECT *
FROM animal
WHERE numr IN (
    SELECT numr
    FROM propriétaire
    WHERE villep = 'Paris'
);
```

Une sous-requête utilisée par IN doit être entre parenthèses. / An IN subquery must be parenthesized.

12.

```
SELECT *
FROM consultation
WHERE prix BETWEEN 200 AND 300;
```

BETWEEN utilise AND, et les bornes sont incluses. / BETWEEN uses AND, and both bounds are inclusive.

13.

```
ALTER TABLE assistant
ADD mail VARCHAR(70);
```

La commande complète est ALTER TABLE. / The complete command is ALTER TABLE.

14.

```
ALTER TABLE assistant
CHANGE nomas nom_assistant VARCHAR(20);
```

CHANGE exige le nouveau nom et la définition complète. / CHANGE requires the new name and full definition.

15.

```
ALTER TABLE assistant
DROP FOREIGN KEY fk_assistant_veto;
```

En MySQL, une FK se supprime avec DROP FOREIGN KEY. / In MySQL, drop an FK with DROP FOREIGN KEY.

16.

```
INSERT INTO assistant (numas, nomas, tel)
VALUES (1, 'PEMBA', '0606060606');
```

Le nombre et l'ordre des valeurs doivent correspondre aux colonnes. / Values must match the number and order of columns.

17.

```
UPDATE assistant
SET tel = '0606060606'
WHERE numas = 1;
```

UPDATE exige SET avant les affectations. / UPDATE requires SET before assignments.

18.

```
DELETE FROM assistant
WHERE numas = 1;
```

DELETE FROM ne prend pas *. / DELETE FROM does not use *.

19.

```
CREATE TABLE eleve (
  idel INT NOT NULL AUTO_INCREMENT,
  nomel VARCHAR(30),
  CONSTRAINT pk_eleve PRIMARY KEY (idel)
);
```

Séparez les définitions par des virgules et mettez la colonne de PK entre parenthèses. / Separate definitions with commas and parenthesize the PK column.

20.

```
SELECT r.numr, r.nomr, COUNT(a.numa) AS nb_animaux
FROM race AS r
LEFT JOIN animal AS a ON a.numr = r.numr
GROUP BY r.numr, r.nomr
ORDER BY nb_animaux DESC;
```

Avec LEFT JOIN, COUNT(a.numa) renvoie 0 pour une race vide ; COUNT(*) renverrait 1. / With LEFT JOIN, COUNT(a.numa) returns 0 for an empty breed; COUNT(*) would return 1.

17.5 Drill express : reconstruire la charpente / Speed drill: rebuild the skeleton

Sans regarder la fiche, recopiez cinq fois la charpente ci-dessous. À chaque répétition, remplacez les pointillés par une vraie table, de vraies colonnes et au moins un alias.

Without checking the guide, rewrite the skeleton below five times. Each time, replace the placeholders with a real table, real columns, and at least one alias.

```
SELECT expression AS alias_colonne
FROM table AS alias_table
WHERE condition_sur_les_lignes
GROUP BY colonnes_non_agregees
HAVING condition_sur_les_groupes
ORDER BY alias_colonne DESC
LIMIT nombre;
```

Toutes les clauses ne sont pas obligatoires. Elles doivent toutefois rester dans cet ordre lorsqu'elles sont présentes.
Not every clause is mandatory. When present, however, they must remain in this order.

18. Examen blanc / Mock exam

Temps conseillé : 35 minutes. Barème : 20 points. Ne regardez pas la correction avant la fin.

Suggested time: 35 minutes. Marking: 20 points. Do not read the solution section before finishing.

Points	Question FR	English subtext
1 pt	Afficher le nom et le tatouage des animaux tatoués, avec les alias animal et tatouage.	Display the name and tattoo of tattooed animals using aliases animal and tatouage.
1 pt	Afficher les propriétaires de Paris ou Lyon dont l'email est connu.	Display Paris or Lyon owners whose email is known.
1 pt	Afficher les médicaments dont le nom contient entre 13 et 18 caractères.	Display medicines whose name contains 13 to 18 characters.
2 pts	Afficher nomm, prix_ht, tva et prix_ttc arrondis à deux décimales. TVA 20 %.	Display nomm, prix_ht, tva, and prix_ttc rounded to two decimals. VAT 20%.
2 pts	Afficher le nombre d'animaux par race représentée, du plus grand au plus petit.	Display animal count per represented breed, highest first.
2 pts	Afficher les propriétaires ayant au moins deux animaux.	Display owners with at least two animals.
2 pts	Afficher le ou les médicaments les moins chers.	Display the cheapest medicine or medicines.
2 pts	Afficher les animaux sans consultation avec NOT EXISTS.	Display animals with no consultation using NOT EXISTS.

Points	Question FR	English subtext
2 pts	Afficher toutes les races, y compris les races sans animal, avec leur nombre.	Display all breeds, including empty breeds, with their count.
2 pts	Ajouter une colonne diplome VARCHAR(25) de valeur par défaut licence à assistant.	Add a diplome VARCHAR(25) column defaulting to licence to assistant.
3 pts	Ajouter numv à assistant puis créer une FK nommée fk_assistant_veterinaire vers veterinaire(numv).	Add numv to assistant, then create named FK fk_assistant_veterinaire to veterinaire(numv).

18.1 Correction de l'examen blanc / Mock exam solutions

1.

```
SELECT noma AS animal, tatouage AS tatouage FROM animal WHERE tatouage IS NOT NULL;
```

2.

```
SELECT * FROM proprietaire WHERE villep IN ('Paris','Lyon') AND email IS NOT NULL;
```

3.

```
SELECT * FROM medicament WHERE CHAR_LENGTH(nomm) BETWEEN 13 AND 18;
```

4.

```
SELECT nomm, prixm AS prix_ht, ROUND(prixm*0.20,2) AS tva, ROUND(prixm*1.20,2) AS prix_ttc FROM medicament;
```

5.

```
SELECT numr, COUNT(*) AS nb_animaux FROM animal GROUP BY numr ORDER BY nb_animaux DESC;
```

6.

```
SELECT nump, COUNT(*) AS nb_animaux FROM animal GROUP BY nump HAVING COUNT(*)>=2;
```

7.

```
SELECT * FROM medicament WHERE prixm=(SELECT MIN(prixm) FROM medicament);
```

8.

```
SELECT a.* FROM animal AS a WHERE NOT EXISTS (SELECT 1 FROM consultation AS c WHERE c.numa=a.numa);
```

9.

```
SELECT r.numr,r.nomr,COUNT(a.numa) AS nb_animaux FROM race AS r LEFT JOIN animal AS a ON a.numr=r.numr GROUP BY r.numr,r.nomr ORDER BY nb_animaux DESC;
```

10.

```
ALTER TABLE assistant ADD diplome VARCHAR(25) DEFAULT 'licence';
```

11.

```
ALTER TABLE assistant ADD numv INT NOT NULL;
ALTER TABLE assistant ADD CONSTRAINT fk_assistant_veterinaire FOREIGN KEY(numv) REFERENCES veterinaire(numv);
```

19. Fiche ultra-courte de dernière minute / Last-minute cram sheet

À relire juste avant l'épreuve. / Read immediately before the exam.

One-page mechanical checklist.

Ordre canonique / Canonical order

```
SELECT ...  
FROM ...  
WHERE ...  
GROUP BY ...  
HAVING ...  
ORDER BY ...  
LIMIT ...;
```

Alias

```
SELECT t.colonne AS alias_colonne  
FROM table AS t  
ORDER BY alias_colonne;
```

NULL

```
IS NULL | IS NOT NULL | IFNULL(colonne, 0)
```

Agrégats

```
COUNT(*) | COUNT(col) | COUNT(DISTINCT col)  
SUM | AVG | MIN | MAX  
GROUP BY ... HAVING COUNT(*) > n
```

Sous-requêtes

```
WHERE x = (SELECT MAX(x) FROM t)  
WHERE id IN (SELECT id FROM t2 WHERE ...)  
WHERE NOT EXISTS (SELECT 1 FROM t2 WHERE t2.id=t1.id)
```

Jointures minimales

```
FROM a AS x INNER JOIN b AS y ON y.id=x.id  
FROM a AS x LEFT JOIN b AS y ON y.id=x.id
```

LDD

```
DROP DATABASE IF EXISTS b; CREATE DATABASE b; USE b;  
CREATE TABLE t (... CONSTRAINT pk_t PRIMARY KEY(id));  
ALTER TABLE t ADD col VARCHAR(20);  
ALTER TABLE t MODIFY col VARCHAR(40);  
ALTER TABLE t CHANGE col nouveau VARCHAR(40);  
ALTER TABLE t DROP COLUMN nouveau;
```

Pièges mécaniques / Mechanical traps

- COUNT(*) et non COUNT(). / Use COUNT(*), not COUNT().
- IS NULL et non = NULL. / Use IS NULL, not = NULL.
- Parenthèses autour des OR mélangés avec AND. / Parenthesize OR when mixed with AND.
- WHERE filtre les lignes ; HAVING filtre les groupes. / WHERE filters rows; HAVING filters groups.
- Toute colonne non agrégée du SELECT doit être dans GROUP BY. / Every non-aggregate SELECT column should be in GROUP BY.
- GROUP_CONCAT(col SEPARATOR ' ', '). / Use GROUP_CONCAT(col SEPARATOR ' ', ').
- Alias de table dans FROM ; alias de colonne dans SELECT. / Table aliases in FROM; column aliases in SELECT.
- Tester UPDATE/DELETE par un SELECT avant exécution. / Test UPDATE/DELETE with SELECT first.
- Pour supprimer une FK MySQL : DROP FOREIGN KEY nom. / To drop a MySQL FK: DROP FOREIGN KEY name.